Embedded Control Design Contest Results p. 26 • MCU Mathematics p. 60 • Tackle a Tough Design Project p. 76

www.circuitcellar.com

CIRCUIT CELLAR

THE MAGAZINE FOR COMPUTER APPLICATIONS

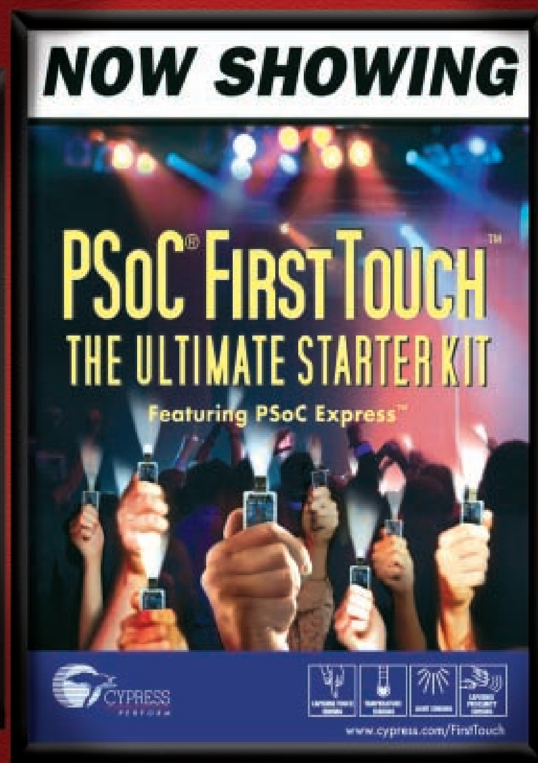
#212 March 2008

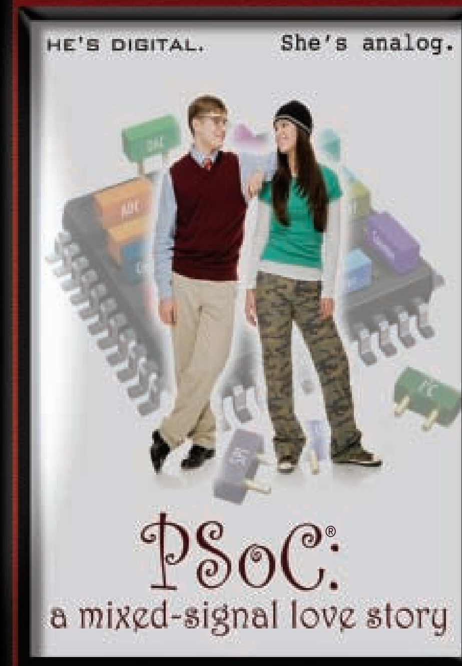
ROBOTICS

03>

7 25274 75349 9

$4.95 U.S. ($5.95 Canada)





It's showtime.

Get Your Ticket to the Ultimate Embedded Design Starter Kit.

Fixed-function microcontrollers had their share of the spotlight. It's time for a revolutionary—and simplified—approach to embedded application development. Get a Cypress PSoC® FirstTouch™ Starter Kit now and discover how much PSoC mixed-signal arrays—powerful, programmable analog and digital blocks, embedded memory and a fast MCU—shorten your time-to-market. This kit includes the easy-to-use PSoC Express™ visual embedded system design tool, and gives you embedded designs you can evaluate right out of the box. Get yours and step into the spotlight today.

CAPSENSE TOUCH SENSING | TEMPERATURE SENSING | LIGHT SENSING | CAPSENSE PROXIMITY SENSING

Includes four ready-to-use mixed-signal applications on a single platform.

Buy your PSoC FirstTouch Ultimate Starter Kit now at:
www.cypress.com/go/FirstTouch

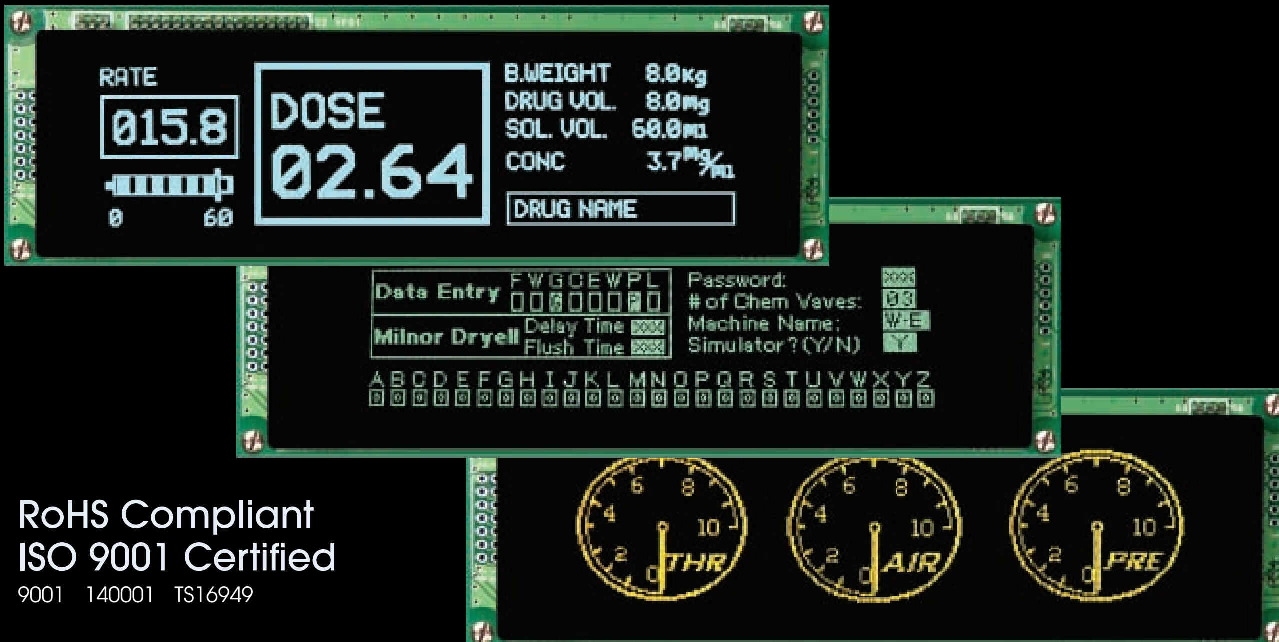
Buy Now
$29.95*

CYPRESS
PERFORM

Noritake

Vacuum Fluorescent Displays vfd

Graphic Display Value
Approaching Character Display Prices
A Display You Can Read From Any Angle



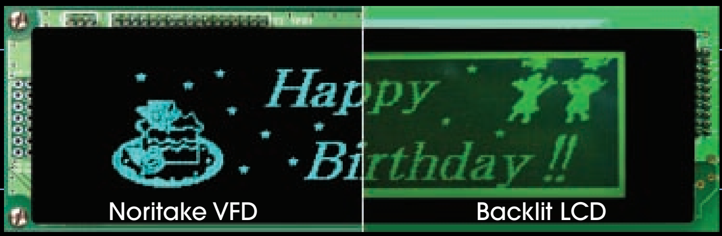
Noritake 7000 Series

RoHS Compliant
ISO 9001 Certified
9001 140001 TS16949

MORE
BUILT-IN FEATURES THAN YOU MAY EVER NEED...

- Up to 4 Independent Windows with Individual Control
- Many Int'l Font Sets with 16 User-Defined Characters
- Easily Combine Text & Graphics on One Screen
- Horizontal & Vertical Scrolling
- Proportional Fonts & Font Magnification
- 8 Levels of Brightness Control
- Built-in Screen Savers
- Many Filters Available to Match Application
- Development Kit Available

DON'T COMPROMISE.
Your application deserves
VFD quality ...



Noritake VFD Backlit LCD

www.noritake-elec.com/51m

Noritake Co., Inc. 2635 Clearbrook Dr., Arlington Heights, IL 60005 phone 1-800-779-5846 e-mail electronics@noritake.com

itron

Over 100 Years in Business

TASK MANAGER

Design, Program, & Prosper

This isn't your average Robotics issue. When we began planning this edition of *Circuit Cellar* several months ago, we set out to put together an issue that would feature a diverse set of articles that would touch on the topics of both hardware design and software development. We could have easily packed the issue with six or seven feature articles about projects like simple motor control applications and line-following robots, but what good would that have done for engineers who are hungry for information about designing and programming interesting real-world applications?

This month, we deliver articles about useful projects that you can easily design and program at your workbench. If you start now, you can have one or two of them up and running by early April.

Want your next robotics project to run smoothly? First, check out why Daniel Ramirez uses the Ada95 programming language for many of his robotics applications (p. 14). Next, jump to page 52, where Monte Dalrymple describes a peripheral chip for low-level functions. He makes a strong case for using the Rabbit Semiconductor I/O (RIO) device in robotics and motion-control projects.

I first introduced you to Miguel Sanchez's innovative vertical plotting system back in September 2007 (Issue 206). Remember the video posted at www.youtube.com/watch?v=VmB14M78CWU? Not long after Miguel showed me the video, we began planning an article about the design. This month, you can learn how it was built and programmed (p. 30). You can build a similar plotter to work in private or industrial settings.

On page 36, a team of designers from Camosun College describes its recent contribution to a flight control system for an RC helicopter. In this article, the team explains how it designed a functional communications system that can be tweaked to work efficiently in a variety of aircraft systems. The team covers the hardware, software, and firmware.

On page 43, Chris Coulston describes the exciting new motion control design that he calls the "Do-It-Yourself Wii." The system enables him to interact with graphics programs on a PC in the same way that gaming enthusiasts use a Wii controller to play video games. In this series of articles, Chris describes how he developed the hardware and software for the system.

In the issue's last feature article, Kripasagar Venkat describes efficient multiplication and division techniques for microcontrollers with no hardware multiplier (p. 60). Although the article is aptly titled "Efficient Micro Mathematics," it could have been called "MCU Math Made Easy."

Our columnists Jeff Bachiochi, George Martin, and Tom Cantrell round off the issue with interesting articles about using ARM processor power to monitor the heavens (p. 69), working with C language when troubleshooting a real design problem (p. 76), and finding the right 32-bit chips for next-generation applications (p. 80).

Finally, note that the winners of the Microchip 16-Bit Embedded Control 2007 Design Contest are listed on page 26. Congratulations to all of the winners!

cj@circuitcellar.com

C. Abate

CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

MANAGING EDITOR
C. J. Abate

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
Jeff Bachiochi
Ingo Cyliax
Robert Lacoste
George Martin
Ed Nisley

NEW PRODUCTS EDITOR
John Gorsky

PROJECT EDITORS
Gary Bodley
Ken Davidson
David Tweed

ASSOCIATE EDITOR
Jesse Smolin

CHIEF FINANCIAL OFFICER
Jeannette Ciarcia

MEDIA CONSULTANT
Dan Rodrigues

CUSTOMER SERVICE
Debbie Lavoie

CONTROLLER
Jeff Yanco

ART DIRECTOR
KC Prescott

GRAPHIC DESIGNER
Carey Penney

STAFF ENGINEER
John Gorsky

ADVERTISING
860.875.2199 • Fax: 860.871.0411 • www.circuitcellar.com/advertise

PUBLISHER
Sean Donnelly
Direct: 860.872.3064, Cell: 860.930.4326, E-mail: sean@circuitcellar.com

ADVERTISING REPRESENTATIVE
Shannon Barraclough
Direct: 860.872.3064, E-mail: shannon@circuitcellar.com

ADVERTISING COORDINATOR
Valerie Luster
E-mail: val.luster@circuitcellar.com

Cover photography by Chris Rakoczy—Rakoczy Photography
www.rakoczyphoto.com

PRINTED IN THE UNITED STATES

CONTACTS

SUBSCRIPTIONS
Information: www.circuitcellar.com/subscribe, E-mail: subscribe@circuitcellar.com
Subscribe: 800.269.6301, www.circuitcellar.com/subscribe, Circuit Cellar Subscriptions, P.O. Box 5650, Hanover, NH 03755-5650
Address Changes/Problems: E-mail: subscribe@circuitcellar.com

GENERAL INFORMATION
860.875.2199, Fax: 860.871.0411, E-mail: info@circuitcellar.com
Editorial Office: Editor, Circuit Cellar, 4 Park St., Vernon, CT 06066, E-mail: editor@circuitcellar.com
New Products: New Products, Circuit Cellar, 4 Park St., Vernon, CT 06066, E-mail: newproducts@circuitcellar.com

AUTHORIZED REPRINTS INFORMATION
860.875.2199, E-mail: reprints@circuitcellar.com

AUTHORS
Authors' e-mail addresses (when available) are included at the end of each article.

CIRCUIT CELLAR®, THE MAGAZINE FOR COMPUTER APPLICATIONS (ISSN 1528-0608) is published monthly by Circuit Cellar Incorporated, 4 Park Street, Vernon, CT 06066. Periodical rates paid at Vernon, CT and additional offices. **One-year (12 issues) subscription rate USA and possessions $23.95, Canada/Mexico $34.95, all other countries $49.95. Two-year (24 issues) subscription rate USA and possessions $43.95, Canada/Mexico $59.95, all other countries $85.** All subscription orders payable in U.S. funds only via Visa, MasterCard, international postal money order, or check drawn on U.S. bank. **Direct subscription orders and subscription-related questions to Circuit Cellar Subscriptions, P.O. Box 5650, Hanover, NH 03755-5650 or call 800.269.6301.**

Postmaster: Send address changes to Circuit Cellar, Circulation Dept., P.O. Box 5650, Hanover, NH 03755-5650.

KEIL™
An ARM® Company

Microcontroller Development Tools

ARM Microcontroller Solution

ARM Powered® Microcontrollers – available from many silicon vendors; offer high computing performance along with rich peripherals. Turn ARM Microcontrollers into your solution for cost-sensitive powerful applications – **with Keil Development Tools**.

C/C++ Development Kit

The RealView **Microcontroller Development Kit** (MDK) is the complete software development environment for ARM7/9 and Cortex-M1/M3.

MDK is easy to learn and use, yet powerful enough for the most demanding embedded ARM application. The integrated Device Database® simplifies tool configuration and includes more than 250 ARM Powered Microcontroller variants for your embedded project.

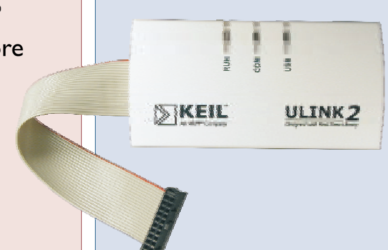
RealView® Microcontroller Development Kit

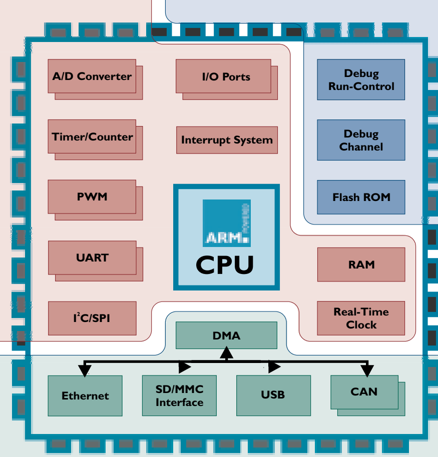
- RealView C/C++ Compiler
- RTX RTOS Kernel Library
- µVision Device Database & IDE
- µVision Debugger & Analysis Tools
- Complete Device Simulation

Examples and Templates

JTAG Debugger

ULINK2® connects to the JTAG or 2-wire debug interface and supports on-the-fly debugging and Flash programming.





RealView® Real-Time Library

- RTX RTOS Source Code
- TCPnet Networking Suite
- Flash File System
- USB Device Interface
- CAN Interface

Examples and Templates

RTOS and Middleware

The RealView **Real-Time Library** (RL-ARM) solves the real-time and communication challenges of your ARM project and expands **MDK** with essential components for sophisticated communication and interface peripherals.

Learn more about RealView **MDK**, **RL-ARM**, and **ULINK2**. Download a free evaluation version from www.keil.com/demo or call **1-800-348-8051**.

Cx51

Keil Cx51 is the de-facto industry standard for all classic and extended 8051 device variants. C51 Version 8.5 includes the latest devices such as XC800, ADE7169, and C8051F4xx - F6xx.

More information: www.keil.com/c51

C166

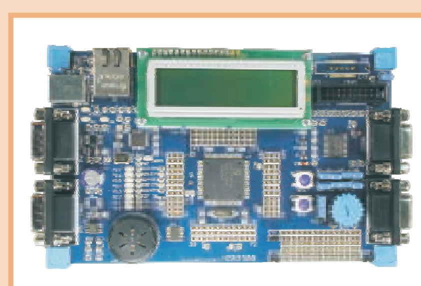
Keil C166 is the complete software development environment for Infineon C166, XC166, XC2000 and ST Microelectronics ST10 with debug and programming support via ULINK2.

More information: www.keil.com/c166

ARM

RealView MDK combines the best-in-class ARM C/C++ Compiler, the genuine Keil µVision IDE/Debugger/Simulator, and the royalty-free RTX RTOS Kernel.

More information: www.keil.com/arm



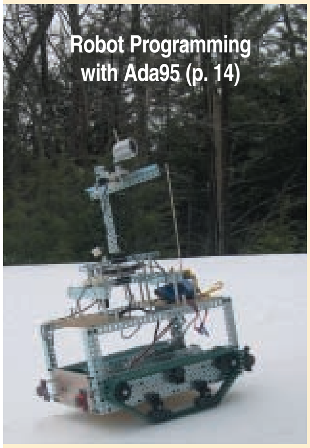
Keil MCB evaluation boards come with code size limited tools and extensive example projects that help you get up and running quickly with your own embedded application.

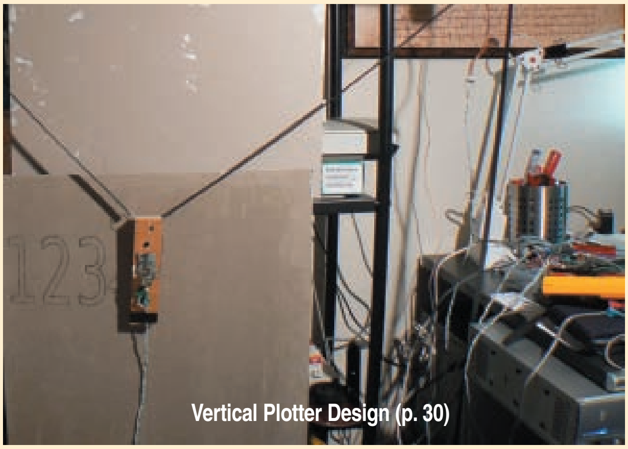
More information: www.keil.com/boards

www.keil.com

March 2008: Robotics

FEATURES


Robot Programming with Ada95 (p. 14)


Vertical Plotter Design (p. 30)


The "DIY Wii" (p. 43)

COLUMNS


A Big Deal in a Tiny Package (p. 80)

DEPARTMENTS


DUEL, WIMP?
This time you are energy depleted history!
Oh No! Not again! When will the boss of "The Waste Land" learn that he can't win?!
My heavy duty Zap-HIM-hard energy rifle feeds on 850 rounds a second. It will blast your blue suit to cinders with zillions of Watts!!
TZJAK ATAK ATAK ATAK ATAK
"I will show you fear in a handful of dust". Have a taste of LOW POWER energy!
TZJAK
ZAP!
ZZPRAKAZZZ
AARGGH! Don't... do this... to me...
Even the smallest of batteries are powerful when you know how to make the most of them!
AARGHHH
AVR®
Fantasi-Fabrikken A/S - Mykle - Wood
AVR® picoPower™

NEW PRODUCT NEWS

Edited by John Gorsky

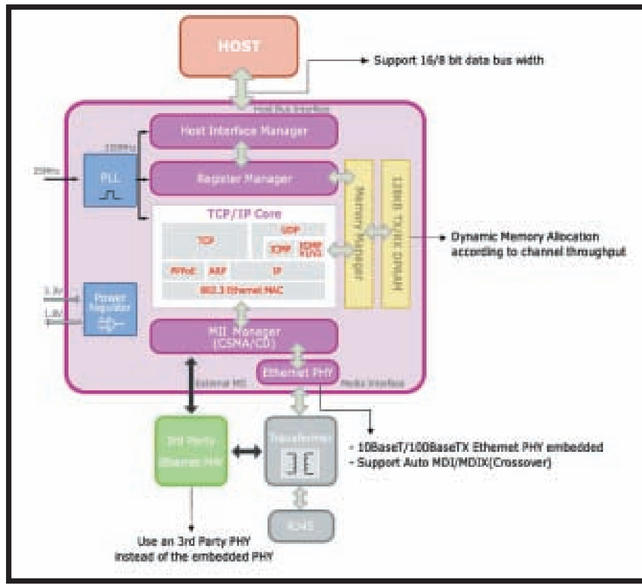
Visit www.circuitcellar.com/npn for more New Product News.

ETHERNET CONTROLLER FOR HIGH-END SYSTEMS

WIZnet has rolled out its newest chip, the **W5300**, which is twice as fast and offers more channels than the currently available W5100. Like its predecessor, the W5300 integrates an Ethernet MAC and PHY as well as a fully hardwired TCP/IP core into an easy-to-use chip. However, it also enables you to use an external third-party PHY rather than the internal one.

The W5300 offers stable speeds up to 50 Mbps, which is enough to watch movies in real-time rather than on a buffered basis. It also supports eight independent socket channels, provides dynamic memory allocation according to channel throughput, and offers a 16- or 8-bit data bus. The W5300 has the capabilities to meet Triple Play Service (Broadband Access, VoIP, and digital broadcasting) needs.

The estimated cost for the W5300 is **$5.99** in 1,000-piece quantities.

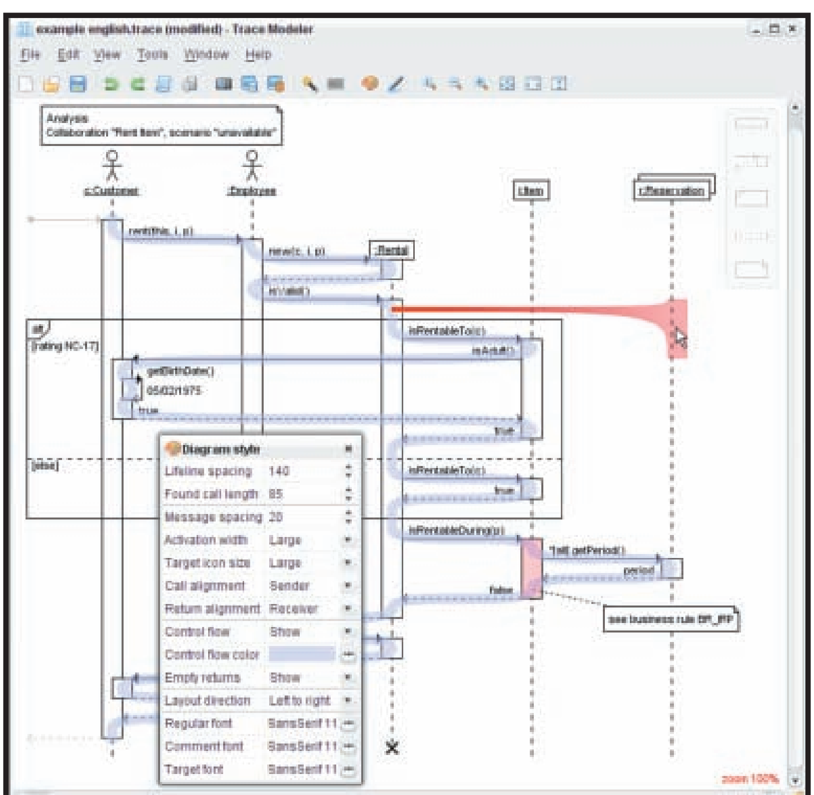


WIZnet, Inc.
www.wiznet.co.kr

EASY-TO-WORK-WITH UML SEQUENCE DIAGRAMS

The **Trace Modeler v1.0** is an easy-to-use and smart UML sequence diagram editor that runs on any platform. It saves developers a tremendous amount of time by combining the immediate and automatic layout of UML sequence diagrams with a simple drag-and-drop interface. The Trace Modeler distinguishes itself by instantly adapting the diagram layout to any changes and guaranteeing attractive and structurally correct diagrams.

The Trace Modeler is designed to optimize developer productivity. It can be used by any development team for brainstorming and prototyping the design of new computer systems or for the documentation of existing ones.



At the heart of its technology lies an understanding of the flow of control in sequence diagrams. This domain knowledge is used not only to determine the proper diagram layout, but also to interpret user actions. This enables the user to quickly change a diagram with simple drag-and-drop gestures, while the Trace Modeler maintains its correctness and pleasing layout. As a result, ideas can be expressed almost instantly with the Trace Modeler.

Diagrams are stored in a simple and readable text-based file format, ideal for versioning systems and file comparison tools. This format makes it easy to generate sequence diagrams with other tools or from running code and visualize them with the Trace Modeler. Diagrams can be exported to various popular graphics formats and the clipboard. It also offers a batch export feature that can be used from the command line for easy integration into any automated documentation process.

The Trace Modeler is extremely portable: it works on all major platforms, requires no installation, and has very accommodating license terms. A single-user license costs **$119** and significant academic and volume discounts are available.

Tracemodeler.com
www.tracemodeler.com

NEW PRODUCT NEWS

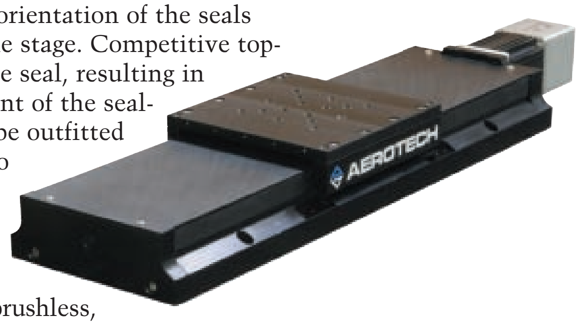
LINEAR BALL-SCREW STAGE

The **ATS165** is a rugged, value-driven linear stage that is ideal for applications requiring debris protection, medium travel, and mid-tier positioning performance. It is similar in design to the ATS115 series, providing additional load-carrying capabilities with larger bearings and a wider cross-section.

The hardcover design provides protection from debris, and the robust aluminum cover is hard-coated to provide a scratch-resistant surface. The side seals keep dirt and particulates off of the stage and protect the bearing surfaces from contamination. The vertical orientation of the seals easily deflects debris away from the stage. Competitive top-seal designs can ingest debris in the seal, resulting in the eventual failure and replacement of the sealing mechanism. The tabletop can be outfitted with an optional brush assembly to remove any particles that collect on the hard cover.

The ATS165 has a NEMA 23 motor flange-mounting interface and can be ordered with a standard brushless, slotless rotary motor installed. The stage can also be ordered without a motor so third-party DC brush, brushless, or stepper motors can be used.

The stage is equipped with a ground 5-mm/rev ball screw that provides 0.5-µm resolution and can be outfitted with English and metric bolt-hole pattern tabletops. A stage-mounted brake option is available to prevent you from back driving the screw when power is removed from the servomotor when the stage is in a vertical orientation.



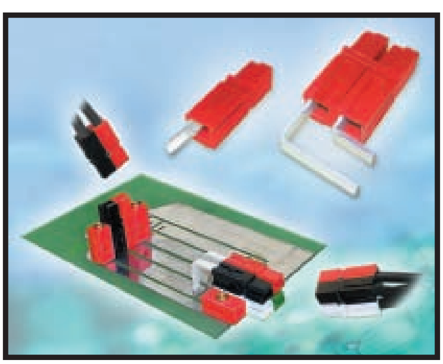
A fold-back kit is also available to effectively reduce the overall stage length for space-constrained applications.

The cost of the ATS165 is dependent on the exact configuration ordered. You may contact Aerotech for a quote.

Aerotech, Inc.
www.aerotech.com

HIGH POWER PCB CONNECTOR

The new **High Power Connector and Contact series** enables high-current connections directly to the PCB. The High Power Connector is an ideal solution for applications that require high-power connections to a PCB.



The High Power Connector and Contact series devices are available pre-assembled in one- or two-row configurations, reducing assembly costs and eliminating accidental connector separation. Connectors and contacts are also available separately, allowing for the customization of connector configurations. The connectors offer high durability and are rated for up to 1,500 cycles under no load and 250 hot plug cycles under a 120-V load.

When used with the APP Powerpole 15/45 finger proof connector housings, the High Power Connector and Contact series provides a reliable and safe wire to PCB connection. The connectors are RoHS-compliant, have a UL rating of 45 A per circuit, and use contacts that are composed of copper alloy with tin plating. The connectors are available in many colors and with multiple accessories that include mounting wings, spacers, and board-mounting staples.

The devices start at around **$1.21** for 1 × 2 vertical connectors in 1,000-piece quantities.

Anderson Power Products
www.andersonpower.com

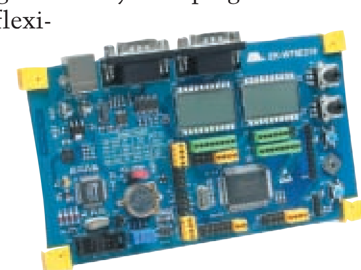
NEW 8-BIT CONTROLLER SERIES

The new **Rich series controllers (W79E217/W79E227/W79E225)** use the Winbond 8051 core, which is capable of executing a single instruction in four clock cycles. The cores are built-in, with 64 KB/64 KB/16 KB of flash memory, and have highly integrated peripheral functionalities (e.g., 10-bit ADC, UART, I^2C, SPI port interfaces, PWM, and internal reset).

Other key features are also integrated into the controllers, including fast execution, high resistance to interference, and industrial specifications. They are applicable to common embedded applications that require high performance and quality features.

The series design provides a typical solution for applications that require small package sizes with sufficient I/O and memory capacities. The series can be applied in a wide variety of general industrial controls, including data loggers, transmission systems, power controls, motor controls, AC controls, and treadmills.

The Rich series provides a full-featured hardware/software development system such as the JTAG hardware simulator, software development system, programmer, and support files. The complete system combination enables the designer to develop products in a short period of time and allows their rapid integration into the market. With the Rich series, in-system program (ISP) functionality can also be used to directly upgrade the system program online, fully realizing the flexibility and expandability of the products.



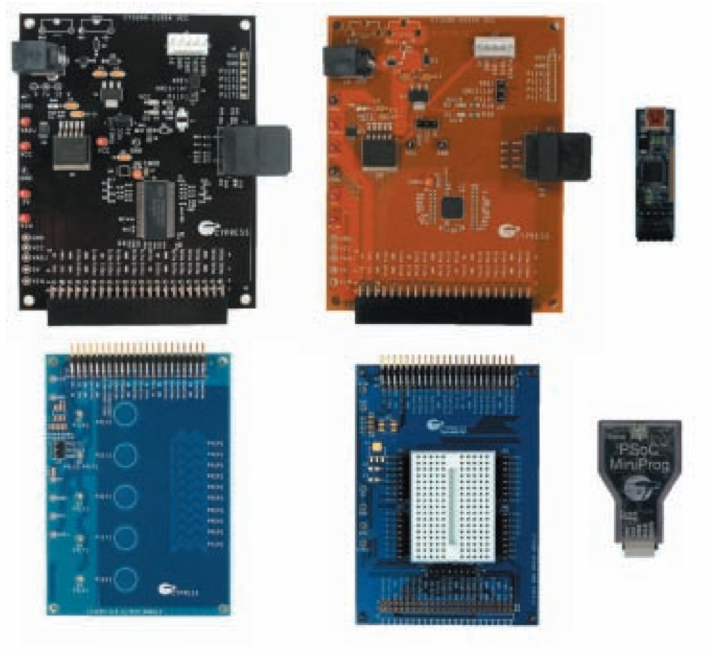
Please contact Winbond for the cost of this new series of controllers.

Winbond Electronics Corp.
www.winbond.com

NEW PRODUCT NEWS

UNIVERSAL CAPSENSE CONTROLLER DEVELOPMENT KIT

The **Universal CapSense Controller Kit** is designed for easy development and debug of any CapSense design. Its modular approach enables both custom and predefined designs to be completed quickly and easily. The development kit comes with predefined control circuitry and plug-in hardware, along with controller boards for the CY8C20x34 and CY8C21x34 PSoC devices. It also includes a breadboard module and a module for implementing up to five buttons and a slider with sample overlays.



The kit works with both PSoC Designer and PSoC Express, and allows for monitoring and tuning of CapSense designs via an I²C-to-USB bridge that is included in the kit. Cypress Semiconductor is unique in offering real-time monitoring and tuning, which can significantly reduce development cycles. The kit supports both the CapSense Successive Approximation (CSA) and Sigma-Delta (CSD) capacitive sensing methods. CSA offers outstanding interference immunity and low-power consumption, making it ideal for portable consumer applications. CSD delivers flawless operation in wet conditions and offers superb temperature response, ideal for white goods and other moisture-sensitive systems.

The kit costs **$158**.

Cypress Semiconductor Corp.
www.cypress.com

NEW PRODUCT NEWS

CAPTIVE HYBRID LINEAR ACTUATOR

The **28000 series** (size 11) double-stack, captive hybrid linear actuator is the latest addition to a continuously growing line of hybrid motors. The compact frame size is only 1.1″ (28 mm) square. The various linear actuator options deliver unmatched capabilities, opening avenues for equipment designers who require performance and endurance in a compact package. Built for long life (several million cycles) and superior performance, the size 11 linear actuator features engineering thermoplastics in the rotor drive nut and a stainless steel Acme lead screw. With this patented design, quiet and maintenance-free operations are ensured. This captive hybrid linear actuator is available in a wide variety of resolutions ranging from 0.000125″ (.003175 mm) per step to 0.002″ (.0508 mm) per step and delivers thrusts of up to 30 lb (133 N) without compromising its long life.

The captive linear actuator is designed with a patented integral anti-rotation mechanism and is especially well-suited for applications in which external anti-rotation methods would be difficult or impossible to employ. Captive linear travel ranges from 0.5″ (12.7 mm) to 2.5″ (63.5 mm). Applications requiring precise positioning and rapid motion are best suited for this motor. Typical applications include medical equipment, semiconductor handling, valve control, X-Y tables, handheld instruments, telecommunications, and many more. Priced aggressively, this product is ideal for incorporation into your next project.

Haydon Switch & Instrument, Inc.
www.hsi-inc.com

ACCURATELY MEASURE TEMPERATURE OVER AN IP NETWORK

The new Q series, **Temperature Module** is a distributed data acquisition module that can monitor up to four digital temperature sensors. The Temperature Module provides an accurate way to measure temperature over an IP network. It can also be used for environmental temperature monitoring and simple control. The module features two internal relays that can be used to control alarm signals, heaters, or coolers.

Like all of the ControlByWeb products, the Temperature Module has a built-in web server so temperatures can be viewed and relays can be controlled using a web browser. The module can also be monitored and controlled using simple XML-formatted text and Modbus/TCP.

The unit can be configured to trigger relays when a set temperature is reached, and it can even control relays in other ControlByWeb products located somewhere else on the network! Configuration of the Temperature Module is easy because it can be fully configured in minutes using a web browser. No configuration software is required.

Standard models of the Temperature Module start at **$107.16** in 100-piece quantities.

Xytronix Research & Design, Inc.
www.controlbyweb.com

NEW PRODUCT NEWS

NEW RABBITCORE FEATURES MEGABYTE CODE SUPPORT

The **RCM4300 series** enables a new generation of applications that use more memory for data and code. The onboard mass storage provides even more performance and easier design than any other alternative in its price range. Software design is supported by a new release of the Dynamic C tools. Dynamic C version 10.21 includes the new Megabyte Code Support (MCS), enabling designers to use over 1 MB of SRAM for shared code and data. Pin-compatible with the complete family of Rabbit 4000-based core modules, the RCM4300 supports twice as much code space compared to any other Rabbit core module, enabling complex embedded applications such as data encryption and security-enabled web servers. The RCM4300 series also provides the capability to implement up to 1 GB of storage using an industry-standard miniSD memory card.

To ease design effort and reduce development time, the RCM4300 development kit is available. The development kit has all of the essentials needed to design a microprocessor-based embedded system with mass storage. The kit includes an RCM4300 with a 512-MB miniSD card, a prototyping board, accessories, and development tools to get design engineers up and running quickly. Along with the industry-proven Dynamic C integrated development software—incorporating an editor, compiler, and in-circuit debugger—there is also the FAT file system familiar to many programmers, RabbitWeb for creating HTML web pages, and Rabbit's Secure Socket Layer utility.

The RCM4300 costs **$80** for 1,000 units. The RCM4310 costs **$69** for 1,000 units. The RCM4300 development kit is **$299**.

Rabbit Semiconductor, Inc.
www.rabbit.com

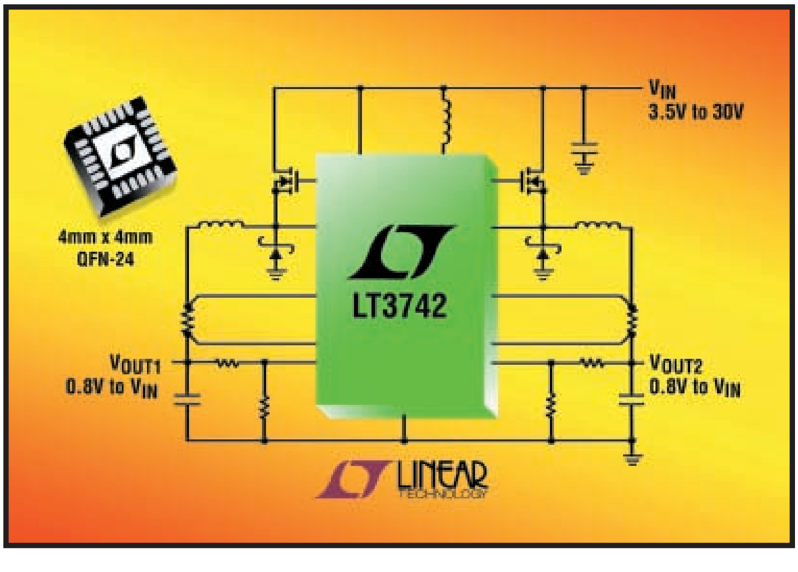
HIGH-OUTPUT VOLTAGE DC/DC CONTROLLER

The **LT3742** is a 100% duty cycle dual-output, step-down switching regulator DC/DC controller that produces output voltages of up to 30 V for applications including stepper motors, industrial controls, automotive, distributed power, and telecom systems. The LT3742 can also be configured with one side as a step-down voltage regulator and the other as a Super Capacitor charger (current source) for capacitor values of up to several farads. This is a useful feature in robotic applications when longer hold-up times are needed or where high peak currents are required.

The LT3742's 100% duty cycle enables the output voltage to be very close to the input voltage, ideal for battery-powered systems. The 4- to 30-V input voltage range enables operation from a number of intermediate bus voltages. A simple step-up DC/DC converter is incorporated to generate its own gate drive voltage and also eliminates the extra parts required to drive the high-side MOSFETs. The onboard gate driver powers a single external N-Channel MOSFET for each output, enabling output currents up to 5 A. A constant 500-kHz operating frequency with current mode control provides fast transient response and easy compensation. Output current sensing is precisely measured by monitoring the voltage drop across a sense resistor and protects the converter during overload and short-circuit conditions. In addition, the LT3742 has adjustable soft-start to control the output voltage ramp time and a good power signal for each output.

The LT3742 is offered in a 4 mm × 4 mm QFN-24 package. One-thousand-piece quantities begin at **$2.30** each.

Linear Technology Corp.
www.linear.com



CIRCUIT CELLAR

Test Your EQ

Edited by David Tweed

Problem 1—The master clock on a GPS satellite is set to 10,229,999.99545 Hz. Why such an "odd" value?

Problem 2—What is the relationship, if any, between the GPS master clock and the GPS microwave carrier frequencies L1 and L2? Why are two different frequencies used?

Problem 3—Both Verilog and VHDL support the concept of compile-time adjustments to the design of a particular module, in addition to the real-time signals that pass into and out of it. What is the specific syntax in each case?

Problem 4—For the data structure commonly known as a "hash" ("associative array" in some scripting languages), what is the difference in overall performance (in order-of-magnitude terms) between storing items into the array and reading them out again?

Contributed by David Tweed

What's your EQ?—The answers are posted at www.circuitcellar.com/eq/
You may contact the quizmasters at eq@circuitcellar.com

Robotics with Ada95

Ada95 is a standardized object-oriented programming language that was developed in the late 1970s by the U.S. Department of Defense. Today, engineers like Daniel use the highly structured language to increase the safety and reliability of their new robotics applications.

What do robots in space, the International Space Station, the Canadian robot arm, the European Arienne rocket, the Boeing 767, the Eurobus jet, and the French Grand Vitesse bullet train have in common? The answer is Ada95, a computer language used to develop life-critical embedded applications. Ada was developed during the late 1970s and early 1980s by the U.S. government for defense applications in an effort to reduce the high life-cycle cost of large software applications. It was also intended to replace older languages, such as COBOL, FORTRAN, and PL/1, which were used by various government agencies.

Ada was developed to fill the need for highly reliable fault-tolerant software. We engineers and robot designers should also be concerned with our robotic inventions that use embedded software. Why? Because they are growing in size and weight and could possibly cause harm or injury to innocent bystanders. The Robot Wars and Tetsujin exoskeleton competitions, where teams design robotic exoskeletons capable of lifting heavy weights, are dangerous events. Just imagine an out-of-control, fire-breathing autonomous battle 'bot. Sounds like a Stephen King movie, doesn't it?

Now you can take advantage of the highly structured Ada95 language for your own purposes because it is freely available on the Internet via GNU/GNAT user-supported sites. An LRM, a complete user manual, and other reference manuals are available at www.adahome.com/Resources/Compilers/GNAT.html. It is easy to get started with Ada95, especially if you already know similar programming languages such as Java, Pascal, Visual Basic, and Visual C++. There are also many references and textbooks on Ada95. I highly recommend J. G. P. Barnes's *Programming in Ada: Plus an Overview of Ada 9X* and "Ada 2005 Rationale" by Intermetrics. Like C++ and Java, Ada95 supports object-oriented programming (OOP) and procedural programming (if you don't want to get into OOP).

What makes the Windows GNAT Ada95 good for the embedded systems developer are the tightly typed variables, tasking, and other structural and safety features. They include: Ada95 packages, generic functions that are similar to C++ templates, types, tagged types and protected records that are similar to C++ classes, ranges, schedulers, tasks that are similar to C++ and Java threads, and Ada exceptions that are similar to the C++ `try`, `catch`, and `throw` keywords.

Real-time multitasking, multiprocessing, and parallel processing are areas that Ada95 handles well. They are also the reason it is used for the development of life-critical software. The Ada95 task instruction with its related delay, select,

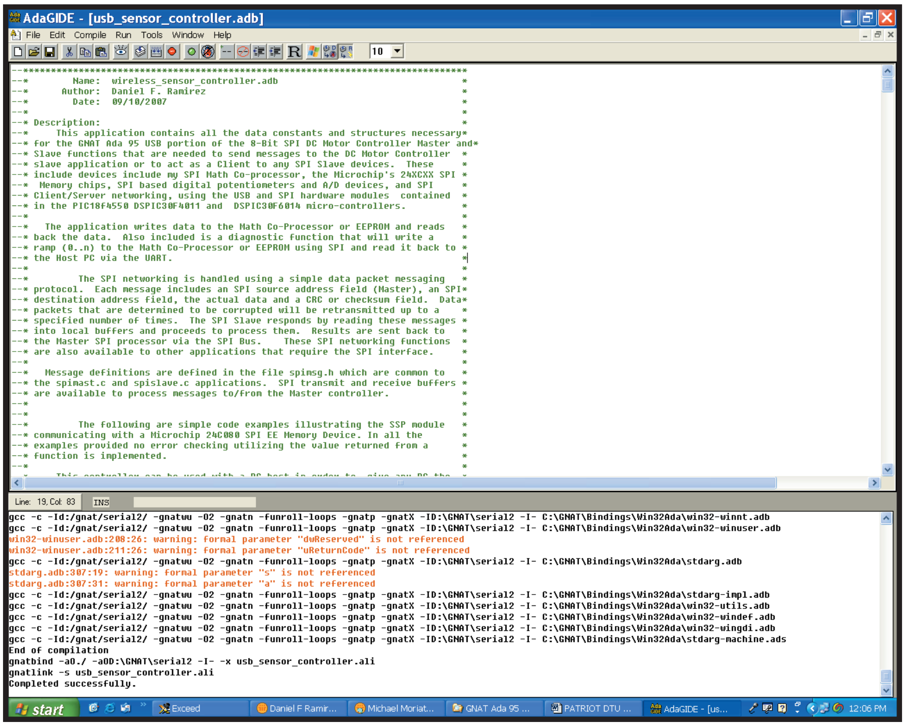


Photo 1—The GNAT IDE is easy to learn and use. It provides menus for editing, compiling, linking, and running your Ada applications.

Listing 1—Here is an example of communication between your Ada95 application and a remote robot at 115,200 bps, 8 data bits, and 1 stop bit. I used the serial port to access wireless telemetry (joystick positions) from a wireless sensor controller using the XBee-PRO wireless UART.

```
--*******************************************************************
--* Initialize_Joystick - Initialize XBEE Pro boards running the
--* Joystick firmware that are connected to the PC or laptop
--*******************************************************************
  procedure Initialize_Joystick (Com_Port : IN OUT Com_Ports.Handle_Type; Name : IN STRING;
          Baud_Rate: Interfaces.C.Unsigned_Long    )     IS
               Joystick_ID : integer := 1;
  begin -- Initialize_Joystick

      -- Open the serial ports for SARD/EVB wireless UART at 115200, N, 8, 1...
      Com_Ports.Open (Com_Port, Name, Baud_Rate, 'N', 8, 1);


  exception
    when E : others =>
      ADA.TEXT_IO.Put_Line(Ada.Exceptions.Exception_Name (E)     & "---" &
      Ada.Exceptions.Exception_Message(E));
          ada.text_io.Put_Line ("Initialize_Joystick: exception 'others' ");

  end Initialize_Joystick;
```

and accept statements and its own deterministic scheduler make it ideal for hard real-time systems that require synchronization. In particular, the protected record structure enables two or more tasks to communicate via shared global data (critical region) without having it corrupted by either task and without the need for semaphores.

Ada95 packages and tagged types encapsulate both data and functions in a similar manner to C++ classes. Packages also control the visibility of objects so some implementation details can be hidden from other packages for information hiding.

Another wonderful feature of Ada95 is bindings. A binding is an interface to another language such as C or assembly using the Ada "pragma interface to C." This is great because it gives your Ada95 applications access to most of the hardware on a PC or laptop. The serial bindings give you access to the serial port. Bindings to the Ethernet are provided via the Winsock bindings that enable you to develop client and server applications. In addition, you can access the mouse via the mouse bindings and a joystick via the joystick bindings for those telepresense and telerobotic applications that you are working on. All of the Win32 bindings and the DirectX graphics bindings necessary for real-time displays and Windows panels are also available.

The language features will help you build more robust, safer, and fault-tolerant robotic applications with my examples as a starting point using AdaCore GNAT Ada95. The Ada features combined with embedded microcontrollers that support the serial communication interface (SCI), the USB interface, and the Ethernet TCP/IP interface (MAC) will help you harness your PC or laptop's gigahertz power for your robot applications.

For designers like me who use Unix, Linux, and LynxOS, there are real-time bindings to POSIX and commercial bindings to WindRiver's VxWorks that can also use GNAT Ada95 for Unix to develop similar applications. Unix and Linux users don't need to be left out from embedded robotics development because of a lack of Windows-specific software tools.

WHY USE Ada95 FOR ROBOTICS?

Designing a robust, fault-tolerant robot requires detailed up-front design work, including modeling, simulation, integration, and testing. This is true of NASA's robots, including the robot arm and interplanetary spacecraft, such as Viking, Explorer, Galileo, and more, which bring us fantastic photos. The Mars exploration rovers, Spirit and Opportunity, provided scientists with evidence

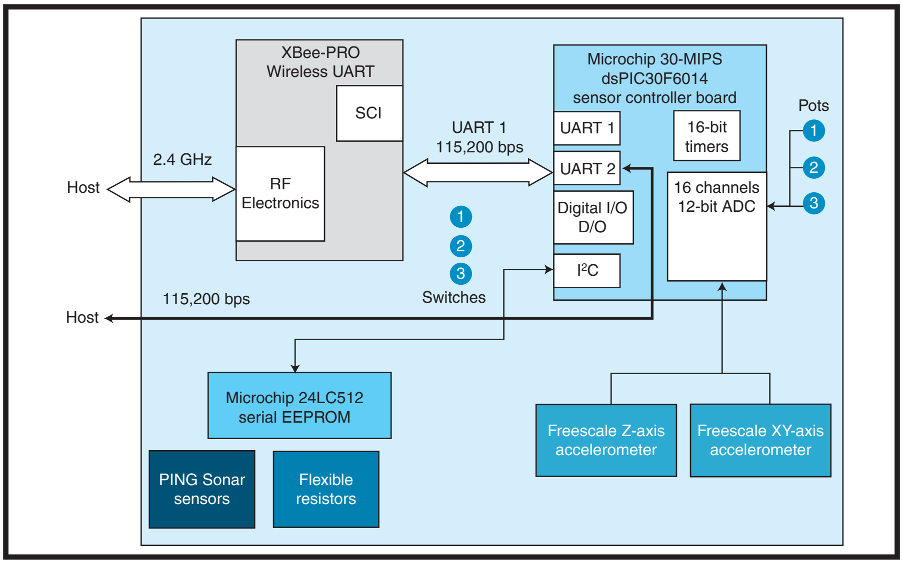


Figure 1—This diagram of the wireless sensor controller shows the main components and interfaces used for the Vex Power Glove.

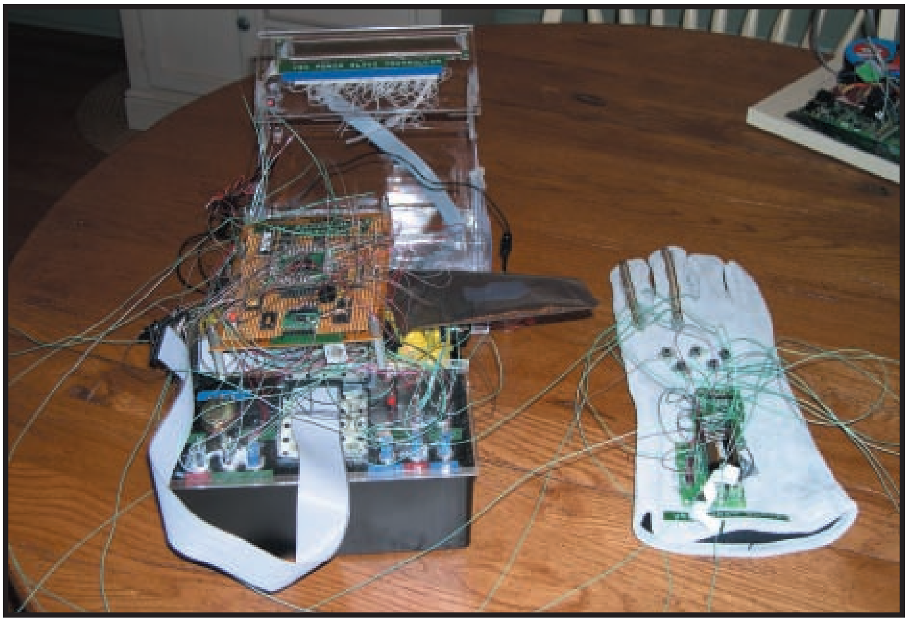


Photo 2—This shows the complete Vex Power Glove system, including the Microchip Technology dsPIC30F6014-based wireless sensor controller and the glove with three flexible resistors and Freescale X, Y, and Z accelerometers (virtual SARD).

of water on Mars. In fact, the International Space Station's robot arm software was written in Ada. The electronic equivalent of an Ada package is the IC. Each IC has specific inputs, outputs, and more.

Features that make Ada95 ideal for robotics include object-oriented methodology, Ada95 packages, type checking, range checking, representation clauses, generic functions, protected records, tasking, deterministic scheduling, and exception handling. Although Java and C++ have similar features, Ada95 enforces type checking, constraint checking, and exceptions. It also has tasking built into the language so it doesn't have to make external system calls or POSIX function calls to schedule threads (similar to tasks).

Ada tasks make developing robotic applications—such as collecting sensor data, checking for objects (obstacle detection), and generating motor commands—easier than using a finite state machine (FSM) or a linear control loop. For example, a sensor collection task and motor control task may run in parallel, with a sensor collection task reading the latest motor positions (encoder counts), while the motor control task sends new PWM motor commands to the motors. Your navigation task determines the next move. Using serial (linear) programming would require a finite state machine (FSM).

Ada's deterministic scheduler makes it easy to schedule tasks and assign them priorities. Passing data between tasks is as easy as a procedure or function call. Simply use accept and select constructs with Ada tasks. Using Ada tasks can make it easier for you to integrate and test the software drivers because it maps well with hardware.

Ada95 protected records provide an easy way for tasks to share data without the need for semaphores and Ada95 tagged types to provide object-oriented functionality similar to C++ and Java. Now you can map Ada tasks and packages to robotic hardware, so adding new hardware and software drivers does not require a complete redesign of your current system. In addition, you also get most of the C/C++ features, including inheritance, function overloading, classes, C++ templates, and tasks.

The key to using Ada95 as a new robot language is to take advantage of its ability to use Win32 libraries via Ada95 bindings and to use its interface to other languages (including C and C++) via the pragma interface function. Now hardware drivers written in C or assembly can be called from Ada95 applications and give them the ability to transmit and receive data via the serial port (COM3 or COM4), via Ethernet TCP/IP, or via the USB interface (USB-HID) bindings. You need these capabilities so you can use your laptop's serial, USB, or Ethernet ports to communicate with your embedded robot applications.

GNAT Ada95 TOOLS

The GNAT IDE is easy to learn and use (see Photo 1). It provides menus for editing, compiling, linking, and running your Ada95 applications. Output is sent to the Ada output window,

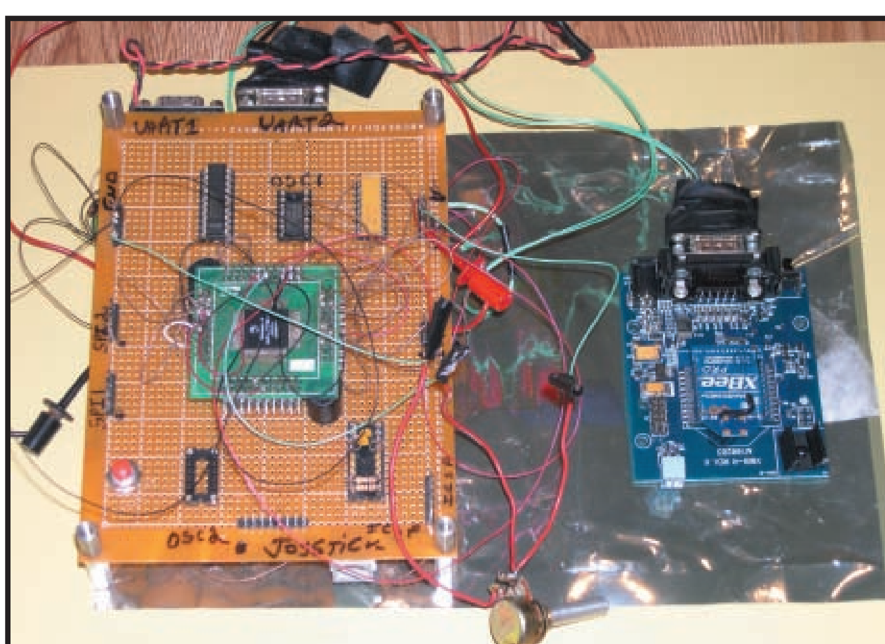


Photo 3—My Vex Power Glove uses a wireless sensor controller connected to a ZigBee XBee-PRO wireless UART that is used to transmit the telemetry to a laptop running an Ada95 data collection application.


www.icoptech.com
Lowest Power Consumption
Most Powerful Performance
Energy Saving and Performance Enhancing can Coexist
New Generation SoC Design
• Fanless Design
• Built-in MTBF Flag Counter
• Onboard 2MB SPI Flash Disk
• Support PCI / ISA / SPI Bus
• Wide Operating Temp. -40 ~+85°C
VSX-6154
Embedded Vortex86SX PC/104 CPU Module
• Vortex86SX CPU 300MHz
• 128MB DDR2 Onboard
• Supports 4S/2USB/VGA/LCD/LAN/GPIO
Low cost 533MHz CPU Module
Fully support ISA Bus
Support DOS, Windows Embedded CE, Windows XP Embedded, Linux
VSX-6115
Embedded Vortex86SX Tiny CPU Module
• Vortex86SX CPU 300MHz
• 128MB DDR2 Onboard
• Supports 4S/2USB/LAN/GPIO
Vega86-6270
Vega86-6270 PC104 Module
• Low Power +5V @ 2.2A,Onboard 128MB SDRAM
• 2D/3D AGP4X Graphics with MPEG2 Accelerator
• Supports 2S/2USB/VGA/LCD/AUDIO/LAN
Vega86-6247
Vega86-6247 PC104 Module
• Low Power +5V @ 2.2A,Onboard 256MB SDRAM
• 2D/3D AGP4X Graphics with MPEG2 Accelerator
• Supports 6S/3USB/VGA/LCD/AUDIO/LAN/CF
ICOP
intelligent control on processor
Email: info@icoptech.com
www.icoptech.com
ICOP Technology Inc.
Los Angeles Office
• Tel: 1-626-444-6666
East Coast Office
• Tel: 856-546-6555
Buffalo Office (NY)
• Tel: 716-650-4018
Windows® Embedded
Gold Partner

which is handy for testing applications inside the IDE. The GNAT gdb (command) or gvd (visual) debugger is also conveniently accessible from the IDE.

One nice thing about the GNAT Ada95 tool suite is that it is absolutely free! Commercial versions of Ada can cost thousands of dollars. If you already have the GNU shareware tools such as GNU C/C++ and the gdb debugger, then you will find the GNAT Ada95 tool suite a welcome edition.

DEBUGGING Ada95 APPLICATIONS

Ada95 provides extensive I/O support using the Ada.Text_IO package (similar in functionality to C's `stdio` library). With the package, you can add trace statements and print data values.

The Ada95 IDE also provides support for using either the GNU gdb debugger or the visual version VDI that is invoked from the GNAT Ada95 IDE. With it, you can examine code, single step, step into, watch variables, and set breakpoints.

Ada95 CODING STYLE

If you look at C/C++ code and the corresponding Ada95 code, you may notice that Ada95 is more readable. This is due to the extra effort the language designers took to make Ada95 easy to maintain. It is self-documenting and bears some resemblance to Pascal. If you remember the Pascal language, you will see some similarities in the keywords.

Listing 1 shows an example of how to communicate between an Ada95 application and a remote robot at 115,200 bps, 8 data bits, and 1 stop bit. The SCI interface on the robot's microcontroller is usually labeled UART0, UART1, SCI0, or SCI1. The exception handler in Listing 1 provides error handling for any runtime exceptions that occur in the routine while using the serial port.

WIRELESS SENSOR CONTROLLER

I used my wireless sensor controller board as a testbed for demonstrating some of Ada95's features, including range checking, tasking, and exception handling using a laptop. As you can see in Figure 1, the controller features a Digi International MaxStream XBee-PRO wireless UART. The UART transmits the telemetry to a laptop running an Ada95 data collection application for my Vex Robotics Power Glove design, which is shown in Photos 2 and 3.

The heart of the Vex Power Glove is a 16-bit Microchip Technology dsPIC30F6014 sensor controller (see Photo 2). The controller provides sensor data to any laptop using the XBee-PRO wireless UART. It can be used to read sensors (voltage, temperature, pressure, and humidity), potentiometers, flexible resistors, and XYZ accelerometers using a sensor task. It also displays them in real time on the laptop's display using a display task. Errors are handled with Ada95 exception handlers. The Ada95 application runs under Windows XP.

I use the Vex Power Glove as a wireless motion input device to teleoperate some of my Vex-based robots, such as the Gilbert III Explorer Robot shown in Photo 4. I took the photo just after a snowstorm last year. The glove transmits finger position and orientation information so it can be processed remotely on a laptop using an Ada95 control application. The application generates motion commands and sends them to the Gilbert III Explorer Robot.

Although I designed the glove for carrying out my own tele-presence and tele-robotics experiments, it can also be used for a total virtual reality experience, such as a control for a PC or laptop-based game that supports standard IBMPC Vex power gloves. It is based on the original Mattel Power Glove, a controller accessory for the Nintendo Entertainment System that was sold during the early 1990s.

WIRELESS MESSAGES

The Ada95 Vex Power Glove application provides robust wireless message processing with CRC/Checksum. The reader can even use the glove's flexible resistors to generate simulated sensor readings that are out of their specified ranges and watch how the GNAT Ada95 application handles the erroneous sensor readings.

Five push button switches provide five digital inputs to the sensor controller. A newer and equivalent dsPIC33F device could be used in place of the dsPIC30F6014 that is even faster (40 MIPS) and provides hardware CRC that could be used to process the wireless messages (telemetry), but I have not tried it for this application.

THE GLOVE IN ACTION

The Vex Power Glove features of a Microchip dsPIC30F6014-based wireless sensor controller that's used to digitize the finger positions (flexible resistors) and the glove's orientation. The XBee-PRO wireless UART and receiver Ada95 application are used to receive glove commands to control a prop, which can be either a laptop or an animation prop. It uses the XBee-PRO receiver and a Freescale Semiconductor MMA1260D Z-axis accelerometer to digitize the glove's current orientation with three of the dsPIC's ADC pins. The glove's orientation is then transmitted back to the laptop (see Figure 1).

The dsPIC firmware digitizes the glove's finger positions by reading the

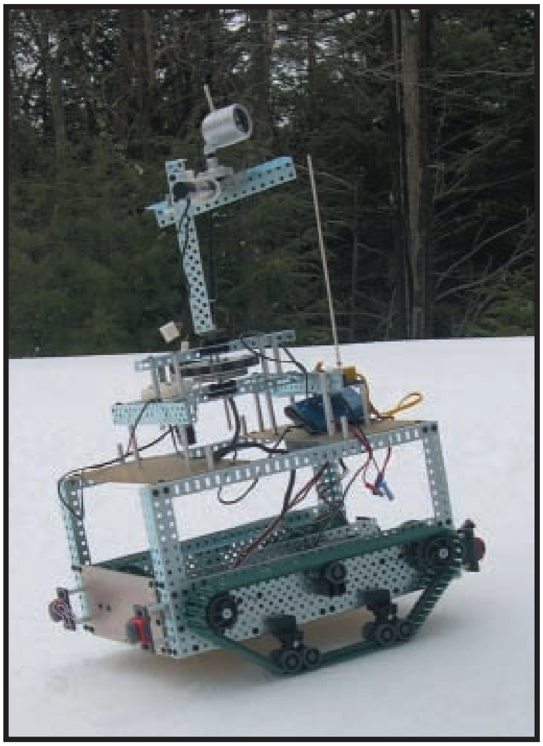


Photo 4—This is my Vex-based Gilbert III robot. I am upgrading the design so I can use the Vex Power Glove as the motion input device. Doing so will enable me to collect data remotely by running an Ada95 data collection application on my laptop.

voltages across five flexible resistors whose resistances vary with the angle percent each finger is bent or flexed. Finger movements are digitized using the dsPIC's 12-bit ADC, and the firmware processes and filters them.

The glove's current orientation relative to the gravity vector is read from the attached Freescale Semiconductor MMA6260Q X/Y-axis accelerometer and the MMA1260D Z-axis accelerometer, which are also digitized by the 12-bit ADC to obtain the orientation angles. The 12-bit ADC is also used to read the pressure sensor attached to the index finger. The 12-bit reading from the ADC for any of the sensors is a value between 0 and 4,095, which can be scaled to PWM motor commands needed for animation.

XYZ ACCELEROMETERS

The MMA6260Q X/Y-axis and MMA1260D Z-axis accelerometers—which are used to obtain the glove's relative orientation—enable you to move associated motors or servos by turning your wrist and adding three degrees of freedom (pitch, yaw, and roll) that can be mapped to servo or motor commands. The accelerometers are read with the 12-bit ADC. With this technology, you can use the Vex Power Glove for video games, input and telerobotics, and virtual reality experiments similar to those carried out in research labs.

An XYZ accelerometer provides the glove orientation (tilt) by measuring the XYZ angles relative to the Earth's gravity vector (to ±1.5 G), which points to the center of the Earth. Although the dsPIC can run with a 3.3-V power supply, I also included a Microchip Technology MCP6S26 six-channel programmable gain amplifier (PGA) for signal conditioning in the design. The MCP6S26 scales the 3.3-V X/Y accelerometer readings to the 5 V required for the dsPIC's ADC because everything else is 5-V based and the voltage reference for the ADC is 5 V. The Z accelerometer is fine because it already uses a 5-V supply. The ADC channels used for the accelerometers are AN12/RB12, AN13/RB13, and AN15/RB15.

The calibration data is saved to nonvolatile memory using Microchip Technology 24LC512 serial EEPROM. It can be retrieved later when it's needed for scaling sensor readings and generating motor commands. A complete schematic diagram of the wireless sensor controller is posted on the *Circuit Cellar* FTP site.

XBee-PRO WIRELESS UART

The 2.4-GHz XBee-PRO ZigBee/802.15.4 RS-232 RF modem (OEM RF module) provides all the benefits of the ZigBee standard in a design that yields three times the range of traditional ZigBee solutions. All of this hardware is provided in a board that measures only 24.38 mm × 27.61 mm, which helps it fit into the enclosure without taking up too much space.

The XBee-PRO wireless UART provides telemetry to a nearby client device (e.g., a laptop, a robot, or another animatronic device). It is configured with the wireless UART application using standard Hayes modem AT commands with the serial port protocol set at 115,200 bps, 8 data bits, 1 stop bit, and no parity. It is connected to the dsPIC30F6014's UART.

The transmitter located on each robot is an XBee-PRO wireless UART connected to the wireless sensor controller. Another XBee-PRO receiver is connected to a Parallax serial servo controller (PSC) mounted on my Hero 2007 Robot, which uses four Vex motors (4-WD) for the motion subsystem (see Photo 5). I used a similar configuration to control the four Vex motors used on the motion subsystem of my telepod robot (see Photo 6). I plan to use Ada95 to process the telemetry received from each of the telepod's sensors using an Ada95 data collection application. I will remotely control the telepod by issuing wireless motor command messages to it.

The advantage of purchasing the kit is that the serial connectors and RS-232 voltage level translations along with the power supply and cables are supplied, including one board that directly connects to the USB port on the laptop. I chose to purchase the kit instead of making the boards for the wireless sensor controller (see Figure 1).

CONTROLLER RANGE

The 100-mW XBee-PRO provides a range of up to 1 mile (1.6 km), and its RS-232 connectivity makes integration simple. The range of this controller exceeds that of a standard Vex controller because the line-of-sight range is the same as the one specified in the ZigBee protocol. The MaxStream XBee-PRO provides the maximum range of 300′ line of sight indoors.

The actual range that the reader gets depends on many factors, including direct line of sight, the antenna length, and the walls and windows that can block the low-power ZigBee signal. The standard antenna is built into the board, but Digi sells other options. Additional XBee-PRO modules can provide greater range and animation control options.

The Digi wireless UART, which can be easily configured using standard AT modem commands, includes a utility to configure the modem channel and data rate, as required

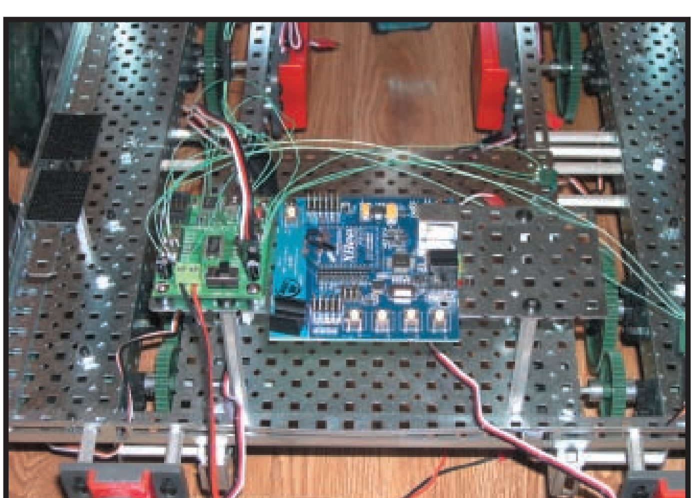


Photo 5—Another XBee-PRO receiver is connected to the Parallax serial servo controller (PSC) mounted on my Hero 2007 robot, which uses four Vex motors (4-WD) for its motion subsystem.

by the application.

USB 2.0 INTERFACE

A Microchip Technology PIC18F4550 USB 2.0 controller provides a convenient method to directly connect embedded controllers to Ada95-based applications running on a laptop or PC by using the USB interface with USB 2.0. This is accomplished by taking advantage of a virtual COM port (VCP), as described in Microchip's USB application notes. It communicates with the dsPIC30F6014 Vex Power Glove controller via the 8-bit SPI bus. You can also purchase a complete USB 2.0 development kit, a PIC18F4550 USB 2.0 controller (PICDEM full-speed USB demonstration board DM163025) that includes a board and all the necessary firmware. I used this board for some of my other USB applications, including a USB high-performance DC motor controller.

The Vex Power Glove can also be directly connected to a PC or laptop as a motion-input device. Of course, this requires the glove to be tethered to a PC or laptop with a USB cable. Doing so provides a convenient way to calibrate, evaluate, and test the glove with resources such as MATLAB and Excel for visualizing the data and analyzing it. You can use GNAT Ada95, Microsoft Visual Basic, or Microsoft Visual C++ for laptop-based glove applications.

Photo 6—I plan to use Ada95 to process telemetry received from each of the telepod's sensors. I will incorporate an Ada95 data collection application. This will enable me to remotely control the telepod by issuing wireless motor command messages.

ETHERNET

Ada95 bindings are available for accessing the Internet and web-based appliances using TCP/IP and sockets. I am in the process of building new Ethernet applications using the new WIZnet W5100 TCP/IP stack for communication and a Linksys wireless router for getting wireless telemetry using the Ada95 TCP/IP bindings to receive telemetry on a laptop. I am also developing new Ada95 applications using the bindings and the information provided by Fred Eady in his recent WIZnet article ("iEthernet Bootcamp: Get Started with the W5100," *Circuit Cellar* 208, 2007).

THE KERNEL

The Windows runtime kernel evolved from a cooperative multitasking operating system to the sophisticated preemptive multitasking and multiprocessing operating system that it is today. Windows is quite an amazing system that can automatically schedule daily, weekly, and monthly tasks such as maintenance, disk defragmentation, space compression, and more. Like the human body, Windows can also repair itself from damage caused by hard disk bad sectors, viruses, worms, and other electronic maladies (e.g., controller failures, SRAM parity errors, bus errors, and voltage spikes). Antivirus software from Norton and MacAfee act in a similar manner to medicines and vaccines used to treat and cure diseases.

Even Unix, Linux, embedded Windows CE, and the Mac operating system are susceptible to these problems, although not to the same extent because hackers do not target them as often. But for Windows, this is a major problem. Windows is an excellent system for current-generation laptops and PCs (over 1 billion copies used worldwide), in part because of the conveniences it affords, including Word, Excel, PowerPoint, Microsoft Schedule, Visual Basic, Visual C++, and many other excellent tools that have been continuously improved over the years following Bill Gates's vision.

While all of this sounds scary, the free version of GNAT Ada95 runs on Windows, Unix, and Linux. (A commercial version sold by Aonix runs under LynxOS.) It is a good platform for non-life-critical applications, but don't try to develop a yacht navigation system with it. Instead, purchase the commercial versions of Ada95 for that purpose. Versions of GNAT Ada95 are also available for LynxOS, Linux, Unix, Red Hat Unix, and other operating systems.

The solution for developing life-critical embedded software is to use an embedded real-time runtime operating system (e.g., IBM Rational, LynxOS, Wind River VxWorks, and Aonix) or use a software development language that includes a runtime as part of the language (e.g., Rational APEX Ada95). Today's option is to use Ada95 with one of the POSIX-compliant runtimes mentioned above. The runtime is tailored to the language and hardware platform (e.g., Intel Pentium, PowerPC, MIPS, SPARK, ARM, or any popular computer architecture).

A problem that could arise with a Java or C++ medical application is if the Windows kernel decides that it is time

to schedule a high-priority internal Windows task. In order to do so, it has to preempt the user's application, which could be a life-critical application used to read a patient's heart rate or EKG during a critical operation. That would make the application miss a scheduled system call inside a real-time task or thread while reading the transducers. The scenario is less likely to occur with a dedicated real-time operating system such as Rational APEX, LynxOS, and VxWorks. The operating systems will also work for other languages such as C++ and Java. Real-time operating systems and run-time kernels have been around for many years and have been continuously improved and optimized to the highest commercial standards for use in embedded enterprise, military, space, and medical applications using a set of standard POSIX-compliant system calls. Commercial variants of Ada95 from IBM Rational, Aonix, and Green Hills have been tailored to POSIX-compliant Ada95 runtimes that work with the Ada95 rendezvous and tasking models to provide robust fault-tolerant performance over a wide variety of life-critical embedded applications.

New applications are being developed for parallel processors and supercomputers used by astronomers and physicists as a replacement for the venerable FORTRAN language that has served the scientific community since the mid-1950s. Ada95 maps well to these applications, including common statements, global memory, and parallelizing expressions.

Ada95 shares many of the features found in other languages including C++ and Java: data abstraction, type checking, range checking, records (C++ structs), object-oriented programming, tagged types (C++ classes), inheritance, generics (C++ templates), multithreading, tasks (multithreading and multitasking), interrupt handling, error handling, exception statements (`catch`, `throw`, and `try`), and raise statements. Although the free version of GNAT Ada95 is not POSIX-compliant and does not have all the optimizations that the commercial variants provide, it may be used for non-life-critical applications and for learning the safety features of the language.

Ada95 VS. OTHER LANGUAGES

C and other languages allow great flexibility when assigning variables of any type, including automatic promotion of types (i.e., the `char` type can be assigned to an `int` type variable with no errors issued from the C compiler). But doing these kinds of operations can lead to run-time errors and system crashes usually caused by unhandled exceptions such as divisions by zero, overflow, underflow, and the square root of negative numbers. In addition, C pointers are even more troublesome when they point to invalid addresses that can cause data to be overwritten (boundary problems).

Other features of Ada that make it an excellent candidate for a robotics language are the multitasking and multiprocessing support that is directly built into it using tasks, a rendezvous mechanism, interrupts, priorities, protected records, and a deterministic scheduler. Ada tasks provide a mechanism for the parallel exe-

cution of applications that map well to robotics hardware (e.g., being able to read and process sensors while simultaneously having a servo task update servo positions). This programming feat can be accomplished in Ada while not having to resort to semaphores or unnecessary system calls. In my application, I can collect telemetry from the Gilbert III Explorer using a sensor collection task. I can then display the collected data on a laptop using a real-time display task, while also polling the joystick and keyboard for operator input.

Best of all, the GNAT Ada95 tool suite is free. It has already been paid for by U.S. tax payers. A new release of Ada 2005 is already going through the ISO approval process and I hope the GNAT 2005 tools will also be available at no cost to the user. The tool suite includes a graphical user interface (IDE), a GNU Ada95 compiler/translator, and GNU C++. In addition, a debugger (gvd) or (gdb) is also available and directly executes from the IDE. Even non-Windows users can take advantage of programming in Ada95 because it also runs in UNIX and LynxOS.

Libraries for accessing the serial port (COM) and the graphics display are also available. There is even a TCP/IP library available that enables Ethernet programming (UDP datagrams), Windows sockets, and more.

Ada95 variable attributes include ranges, types, and subtypes, which provide the basic data types used to increase software safety and reliability. Because range checking or limit checking is performed automatically during the execution of an application by the Ada95 runtime, if variables exceed their predefined limits or ranges, they will generate an Ada95 exception. The exception can be handled by the local package, procedure, or function or propagated to a higher level or handled by the software designer. Other runtime errors that can generate Ada95 exceptions include file I/O errors, invalid user input, math overflow, math underflow, and more. Each of the conditions can be addressed with its own error handler. An example of an Ada95 exception handler is shown in Listing 1. When the handler traps a runtime error, it can either process it or propagate it to the calling routine. It will also display the actual error message and line number where it occurred.

By thoughtfully specifying the minimum, maximum, and nominal values for each variable and giving them their own type or subtype and range values, the software designer adds a level of confidence and increases the reliability of the embedded Ada95 application. The techniques are not limited just to Ada95, because C++, Jovial, and Visual Basic have similar features, although they are not as convenient to code because variable attributes may not be available in some of these languages. Refer to Listing 2 for examples of Ada95 types, subtypes, and ranges.

As a BASIC and C programmer, I love being able to assign one variable to another, regardless of their types and whether it makes any sense. I am guilty of this, and it can lead to severe runtime errors that can cause applications to crash when one variable overwrites another's space. Arrays are particularly

Listing 2—A range has a lower bound, L, and an upper bound, R, for a range L. R is shown in the following examples, which are used to define RC servo commands for a Parallax servo controller (PSC). If the application assigns servo commands outside the specified range, it will raise a `Constraint_Error`. This is a unique Ada capability that can be simulated in other computer languages using additional limit checks.

```
type WORD_TYPE is
      record
        Data_1 : Byte;
        Data_2 : Byte;
      end record;

for WORD_TYPE'SIZE use 16;

-- Declare RC Servo subtypes with ranges here...

-- Use Channels 0..15 with only one PSC board otherwise you can
-- use Channels 0..31
subtype Channel_Type is Byte range 0 .. 15;

-- RC Servo Ramp
subtype Ramp_Type is byte range 0 .. 63;

-- The Servo Position range, (250-1250), corresponds to 0 to 180
-- degrees of servo rotation with each step equaling 2 uSec.
--
--Note: Not all servos are exactly alike. If your servos appear to
--strain when commanded to position 250, you should increase the
--250 up to 260 (or so) to prevent the servo from straining itself.
subtype Parallax_Position_Type is Word range 250 .. 1250;


subtype number_base is integer range 2..16; -- numeric base subtype f
```

susceptible to the errors when the array index exceeds the array limits. C++ and Java support typing and exception handling (`try`, `catch`, and `throw`). Limit checks can be implemented in any language, although they add some overhead. In Ada95, an exception raised is equivalent to the C++ `throw` keyword:

```
raise SENSOR_READ_ERROR;
```

Variable types are easily converted from one type to another by retyping them (`X := float(I);`) as long as the retyped variable being assigned is large enough to hold the information. In this case, `X` is a 32-bit float and `I` is a 32-bit integer, so there is not a problem with the assignment. Refer to the *Circuit Cellar* FTP site for an example of an Ada95 exception handler.

FUTURE APPLICATIONS

Ada95 is currently used to develop life-critical embedded software used in medical devices, aircraft, and aerospace applications, but it can also be used in the future to increase the safety and reliability of commercial and robotics designs. The platform necessary to run Ada95 is now provided by 32-bit microcontroller-based boards that can be used with Windows, Unix, Linux, and LynxOS as their embedded operating system. Virtually any robotics application could benefit from the additional safety features afforded by developing the necessary embedded software using Ada95 (and soon Ada 2005). Although other computer languages have similar features, the tasking, protected records, deterministic scheduling, and real-time constraint checking are unique to Ada95, which make it an ideal candidate for life-critical and parallel-processing applications.

I provided some examples of how you can use Ada95 with today's microcontrollers, including the dsPIC30F6014 16-bit DSP in the wireless sensor controller board that I used to collect data from the Vex Power Glove. Some of my other Ada95 applications include a USB high-performance DC motor controller that enables me to control DC motors from a laptop by sending its messages via the USB interface.[1] I plan to continue using Ada95 as I work on the robotics applications shown in Photos 2, 4, and 6. ◼

Daniel Ramirez is a senior software engineer at Raytheon with over 15 years of experience working on real-time embedded systems. He has a B.S. in Computer Science and an M.S. in Engineering from Northeastern University. His hobbies include watching old movies, antiques, travel, golf, photography, Circuit Cellar *contests, and Vex robotics.*

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2008/212.

REFERENCE

[1] D. Ramirez, "High-Performance Motor Controller," *Circuit Cellar* 206, 2007.

RESOURCES

AdaCore, "GNAT User's Guide," Florida State University Department of Computer Science, www.cs.fsu.edu/~baker/ada/gnat/html/gnat_ugn_toc.html.

J. Barnes, *Programming in Ada: Plus an Overview of Ada 9X*, Addison-Wesley, Boston, MA, 1994.

Intermetrics, Inc., "Ada 2005 Rationale," 1995, www.adacore.com/home/ada_answers/ada_2005.

Special Interest Group on Ada, "Ada 95 Bindings," 2007, www.sigada.org/ada_95/bindings.html.

U.S. Air Force Academy, "Ada Utilities," 2007, www.usafa.af.mil/df/dfcs/bios/mcc_html/ada_stuff.cfm.

Wikipedia, "Ada," http://en.wikipedia.org/wiki/Ada_programming_language.

SOURCES

XBee-PRO wireless UART
Digi International, Inc.
www.digi.com

MMA1260D Accelerometer
Freescale Semiconductor, Inc.
www.freescale.com

dsPIC30F6014 DSP, MCP6S26 PGA, and PIC18F4550 USB 2.0 controller
Microchip Technology, Inc.
www.microchip.com

AdaGIDE
U.S. Air Force Academy
www.usafa.af.mil/df/dfcs/bios/mcc_html/adagide.cfm

R8C/Tiny Improves Efficiency and Adds Intelligence to Motor Systems

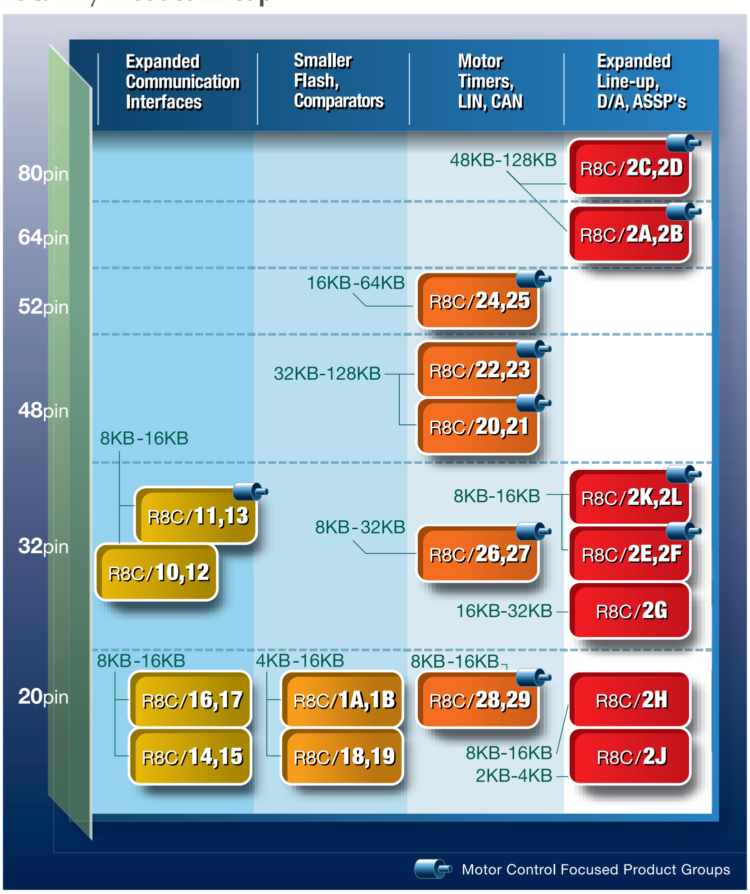
Networked Motors: 40MHz 16-bit Timers and LIN/CAN Modules

Renesas Technology

No.1* supplier of microcontrollers in the world

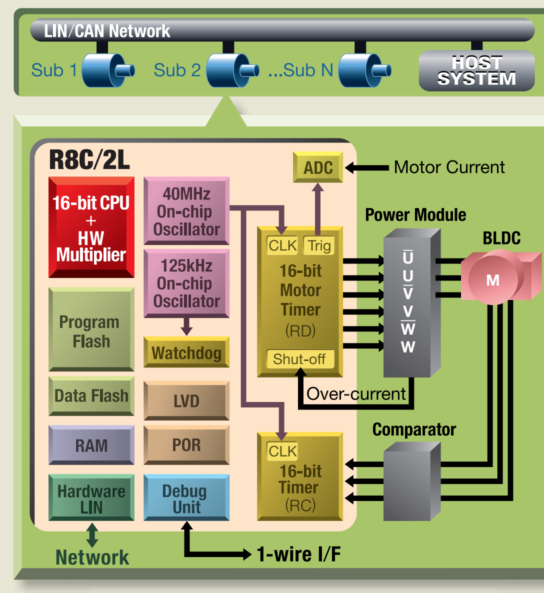
introduces R8C/Tiny Series of microcontrollers for low-end motor control applications. Advanced motor-tuned timers coupled with R8C/Tiny's powerful 16-bit CPU provide the performance demanded by applications ranging from electronic toothbrushes to networked motor arrays in industrial automation.

R8C/Tiny Product Lineup



HOT Products | **R8C/2L**

Ideal for Networked BLDC Motor systems



Top Reasons For Selecting R8C/Tiny's Motor Control Solutions

- **High-Performance**
  - 16-bit CPU core with hardware multiplier
  - Up to three 16-bit timers with 40MHz clock
  - 3.3µs ADC conversion with timer-trigger start option
- **Advanced Motor-Tuned Timers**
  - Up to 6-ch complementary PWM signals with independent compare registers
  - Programmable Dead-Time Control (16-bit)
  - Selectable buffer operation for fast timer reload
  - PWM signal shut-off using external trigger
- **High Component Integration**
  - Integrated LIN and CAN modules for networking
  - Watchdog with dedicated On-chip Oscillator
  - Power-on Reset, Voltage Detection Circuits and Data Flash
- **Outstanding Development Support**
  - Multiple motor control algorithms
  - Demonstration and reference design platforms

*Source: Gartner (March 2007) "2006 Worldwide Microcontroller Vendor Revenue" GJ07168

#1 MCU REACH FURTHER

Get Started Today -

Go online and register to be eligible for a FREE R8C/Tiny Starter Kit *

www.america.renesas.com/ReachR8C/d

*Only available to customers in North and South America

Visit us at Booth #728 & #3028!

Embedded Systems CONFERENCE SILICON VALLEY

April 15-17, 2008

Renesas Technology Corp.

Everywhere you imagine. RENESAS

WINNERS ANNOUNCEMENT

The Microchip 16-Bit Embedded Control 2007 Design Contest gave the world's most talented engineers a chance to test their design skills and learn more about Microchip's 16-bit microcontroller and digital signal controller (DSC) families. Thank you to everyone who participated in this contest.

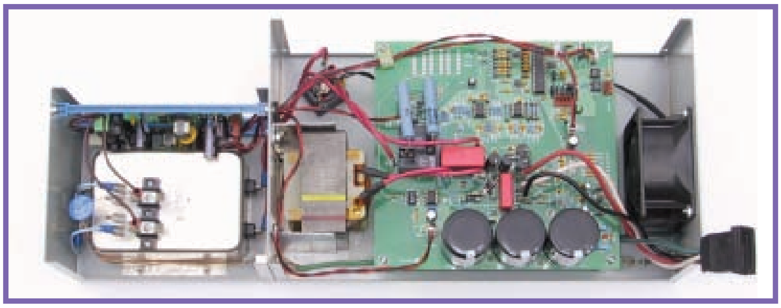
To see the complete projects and more, www.circuitcellar.com/microchip2007/.

FIRST PLACE

Variable Speed Induction Motor Controller

Variable-speed drives for AC induction motors are readily available on the market. But what if you need one for a single-phase motor that operates in the 2-HP range? This innovative controller is the solution. Designed for use with a capacitor-start/capacitor-run motor, it includes active power factor correction (PFC) and inrush current limiting. The motor drive hardware is built around a dsPIC30F2020 DSC. The drive communicates via an optoisolated serial port to a PIC16F687-based interface unit.

Richard Wotiz
U.S.
dick601@mystics.org



"My project is a variable speed drive for a 2-HP single-phase AC induction motor and includes active power factor correction. It has a control port which interfaces with a standard spa control system to control the pump speed. The motor drive hardware is built around a Microchip dsPIC30F2020 digital signal controller. At first I was overwhelmed by the number of DSC part choices, but was able to narrow it down to a part with the exact features I needed. I also found the versatility of the peripherals to be very helpful, even though it made them more challenging to use. In a high-power design such as this, it was hard to know in advance which settings would give the best performance or efficiency, so the many options of the PWM and ADC modules were very helpful to have for last-minute adjustments."
—Richard Wotiz

To see these projects and more, visit www.circuitcellar.com/microchip2007/.

SECOND PLACE & Connected (Communications) Applications

Multifunctional Alarm Clock

Unlike old-school mechanical alarm clocks that you have to set manually, this Internet-connected alarm clock provides three primary features: automatic time setting on power-up, streaming MP3 music, and remote management. The easy-to-use, PIC24FJ64-based clock is connected to an ENC28J60 Ethernet chip, an MP3 decoder chip for streaming music, an organic LED graphical display, and a 24LC512 EEPROM for storing alarm data, fonts, and images.

DJ Delorie
U.S.
dj@delorie.com

"My project is a streaming Internet alarm clock. It has no self-contained music ability, but instead streams MP3s off my office computer. It has the potential to be connected to streaming radio stations. My old alarm clock, which I also built, was dying, and I needed a replacement that let me set the alarm from my office so I wouldn't wake up my wife at night. This project was also an excuse to try out some new technology, like OLED displays and MP3 decoder chips, as well as learn about the PIC24 microcontroller. The PIC24 was easy to prototype with because it came in a standard DIP package. I had the entire design working on a solderless breadboard before committing it to the PCB, and adding support for ICP was trivial. The pin select feature really came in handy, as I needed a lot of peripherals for this project. Without it, I would be forced to use a package with more physical pins."

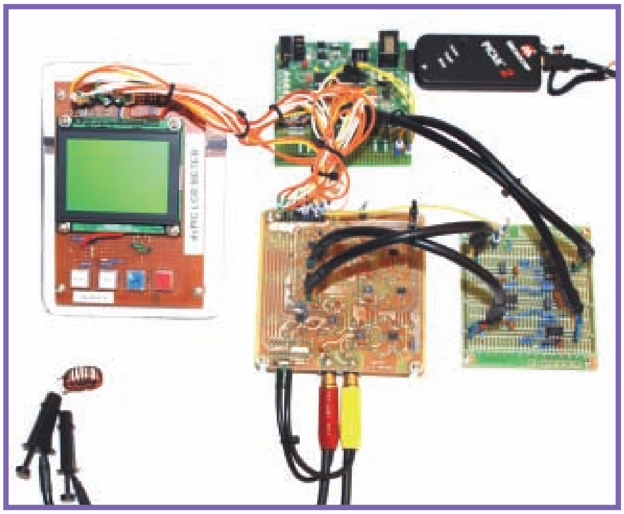
—DJ Delorie

THIRD PLACE

Full-Feature Portable LCR Meter

LCR meters used to be relegated only to professional design labs. Not any more. This portable LCR meter makes it easy for you to analyze the analog performance of virtually any device under test, whether in your lab or on the job. The dsPIC30F4012-based meter uses DDS techniques and DSP methods to condition the resulting voltage and current signals. Its handy user interface and graphic LCD make it easy to operate and read.

Miguel Rusch
Australia
miguel@migdevelopments.com



"I built a prototype for a portable LCR meter. This tool measures the AC impedance of passive devices at a range of frequencies (100 Hz, 1 kHz, and 10 kHz), displaying the measured parameters on an LCD screen alongside a schematic representation of the circuit. It is a tool that's useful for identifying unmarked SMD parts and characterizing hand-wound inductors. The dsPIC processor has a rich set of peripherals to compliment the DSP core. My project obviously required DSP filtering because of the frequency domain measurements; however, without the supporting peripherals, I would have never integrated so much circuitry with a 28-pin device. I made heavy use of the SPI module which communicates with five separate ICs. Additionally, the ADC module enabled extensive configuration, allowing the device to efficiently and simultaneously read the test waveforms and also detect user input from the resistor ladder keypad."

—Miguel Rusch

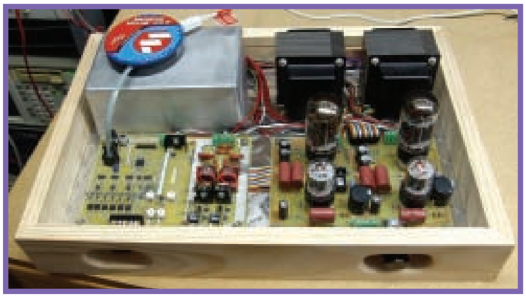
To see these projects and more, visit www.circuitcellar.com/microchip2007/.

Effective Use of SMPS Resources

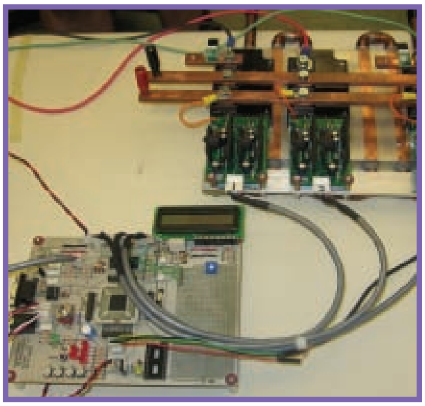
MiniTron Amplifier

The MiniTron is a high-end vacuum tube stereo amplifier with efficiency, distortion, and power output controlled by a dsPIC30F2023. The innovative system features three main subsystems: a vacuum tube amplifier, a DSP controller, and a high-voltage switching power supply. The fully functional amplifier successfully blends its unique circuitry and specialized processing software to precisely match the radically different worlds of high-voltage analog and low-voltage microcontrollers.

George Anderson
U.S.
tech@tubelab.com



"The MiniTron amplifier is a high-end vacuum tube stereo amplifier with distortion, efficiency, and power output enhancements provided by some unique circuitry featuring a dsPIC controller. This project was an opportunity to take vacuum tube technology into the 21st century. I find that Microchip's controllers in general are a bit easier to use than some other microcontroller ICs. I have designed circuits with PICs, but this was my first dsPIC design. The controller section of the design went together smoothly and worked correctly the first time. Microchip's tools and evaluation boards made it possible to test out a good portion of my design before committing to it. There was enough sample code on Microchip's web site to prove out the tracking buck converter on a Microchip evaluation board before building any hardware."
—George Anderson



Effective Use of Motor Control Resources

EV Inverter Project

The Electric Vehicle (EV) Inverter project is a starting point for engineers trying to convert a vehicle with a gas engine to one with an electric motor. The innovative inverter was built using a control board and software based on Microchip Technology's MC-1 development board. It features a dsPIC30F6010A and three dual IGBT modules with gate driver boards. The finished system can power a wide range of AC induction motors.

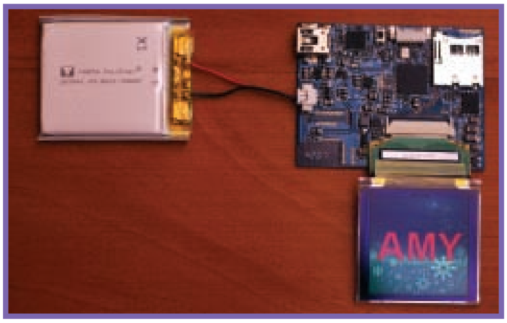
Dena Ponech, Doug Krahn, Adam McIntyre, Tristan Kasmer, and Dan Hall
Canada
ponech.dd@forces.gc.ca

"As the world is currently in the process of advancing alternative sources of energy, the concept of the electric vehicle (EV) conversion is becoming more and more popular. However, anyone interested in this concept will discover that the process of converting a gas-powered vehicle to electric power is fairly cost-prohibitive. Our inverter was constructed using a control board and software based on Microchip's MC-1 development board and three dual IGBT modules with gate driver boards. We chose these development tools to build a basic 100-kW, three-phase inverter in an open-source process, which can be continually improved upon. Microchip's many on-chip resources were easy to use and were an excellent fit for our project."
—Dena Ponech

Use of Peripheral Pin Select

Electronic Mini-Badge

The Electronic Mini-Badge is a PIC16F64GA004-based electronic badge that displays color slideshows. Many electronic badges display scrolling text using a matrix of discrete LEDs. This low-power system features a graphical OLED display and can be easily upgraded to handle animation, video, or RF communication.



"The Electronic Badge project is an electronic version of the standard name badge. I did a simpler version of it before, but I created this version with more functionality. I like the Microchip parts. They are well documented and good support is available in the form of application notes, examples, and discussion forums."
—Jan Szymanski

Jan Szymanski
Australia
janek@bigpond.net.au

To see these projects and more, visit www.circuitcellar.com/microchip2007/.

Effective Use of DSP Technology

NUE-PSK Digital Modem

The dsPIC33F128MC706-based NUE-PSK Digital Modem is a portable PSK31 interface. The stand-alone, battery-operated system enables PSK31 field operation without the use of a PC. The novel system includes two LCDs. LCD-1 is a 4 × 20 character display for transmitting and receiving text data. LCD-2 is a 144 × 32 pixel graphics display for showing the FFT-generated spectrum of the audio passband.

George Heron and Milton Cram
U.S.
n2apb@verizon.net

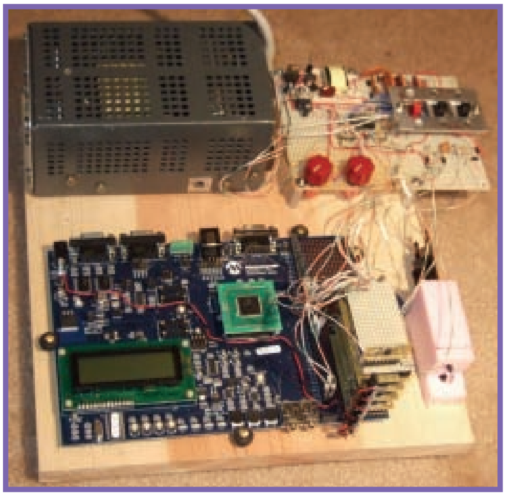
"Our design is a compact digital communications modem for amateur radio operators. Since most digital systems used by amateurs require the use of a computer with sound card, they are not easily used for field operations (e.g., in remote areas). This prompted the design of our low-power, portable modem for use with a single-sideband amateur transceiver, which itself can be quite small and portable (i.e., contained on a single 4† × 4† PC board). Low power consumption, plenty of code and data memory, built-in DSP capability, and low-cost development tools were key requirements for an amateur radio project. The dsPIC33F was an excellent choice."

—George Heron & Milton Cram

Best Use of Bonus Parts

Data Acquisition System for Radiation Measurement

This innovative data acquisition system measures cosmic rays, natural background radiation, and emissions from other radioactive objects. The compact system features dsPIC30F4012 data acquisition processors running in parallel in a "variable time slice" manner. A dsPIC30F6014A controls the data-acquisition processors, gathers the collected data, and formats the data for display.



Peter McCollum
U.S.
saipan59@q.com

"I think of my project as a 'glorified Geiger counter.' The purpose was to build a device for the study of ionizing radiation, and at the same time experiment with data acquisition techniques using Microchip's 16-bit microcontrollers. The system includes an analog front-end that interfaces to three types of radiation sensors, a group of parallel data acquisition processors, and a single supervisor processor that controls the overall operation and interfaces with the user. I enjoy working with the 16-bit family. Because of the small package sizes that are available, it is practical to use them even in smaller, simpler applications that would previously have used an 8-bit MPU."

—Peter McCollum

Use of Smart Sensing Methodologies

Braille Glove

The well-designed Braille Glove system simulates the 26 Braille codes. The system features two gloves that communicate wirelessly. The main board includes a dsPIC33FJ256MC510 microcontroller, which controls four vibration motors, four accelerometers, and a 32-KB SPI serial EEPROM 25LC256. This system provides a unique method for someone to both read and speak using Braille.

Hoa Phan, Vincent Dinh, Tu-Anh Ton, and Nghia Tran
U.S.
hphoa@yahoo.com

To see these projects and more, visit www.circuitcellar.com/microchip2007/.

 by Miguel Sanchez

Vertical Plotter System

Miguel explains how to build a vertical plotter system that can draw on canvas with a pencil or charcoal stick. The compact, portable system consists of a pen holder hanging from pulleys on two stepper motors. In this article, he describes the entire process, from hardware development to programming the system.

Throughout history, painters have used whatever technology was available to help them with their art. It is believed that Johannes Vermeer, like other Dutch masters, used a dark chamber to help him lay down the detailed and realistic impressions on his paintings. The bottom line is that the use of technology is not cheating—or at least that is what my wife told me when I began toying with the idea of building a painting machine to help her with her paintings after spending a weekend serving as her painting machine.

The goal of this project was to build a system that could use a pencil (or charcoal stick) to draw the contour of a scene that you want to paint on a canvas (see Figure 1). Of course, there is still a lot of work for the artist's hand afterwards, but it helps to have this first step done quickly. Plus, it is easily repeatable if you are planning a series of the same image.

One problem with the system was that the size of the painting area was not always the same and I did not have the room for a large painting system. My first idea was to build an XY plotter with a couple of stepper motors, but such a system would require a lot of space and would have to be built for the maximum canvas size. I was looking for alternatives when I discovered a project by some Swiss students called Hektor, a painting machine that paints over a vertical wall with a spray can. A solenoid was used to press the can's nozzle. Although my goal was to use a pencil, I liked the simplicity of a system that hangs from two stepper motors on the two upper corners of a drawing area.

I liked their design, so I built my own. I call it Viktor (it shares the last four letters with Hektor, the project that inspired me to build this one). The system is easy to carry around and you do not need a spare room to install it (see Photo 1). It also requires fewer parts than a standard XY plotter.

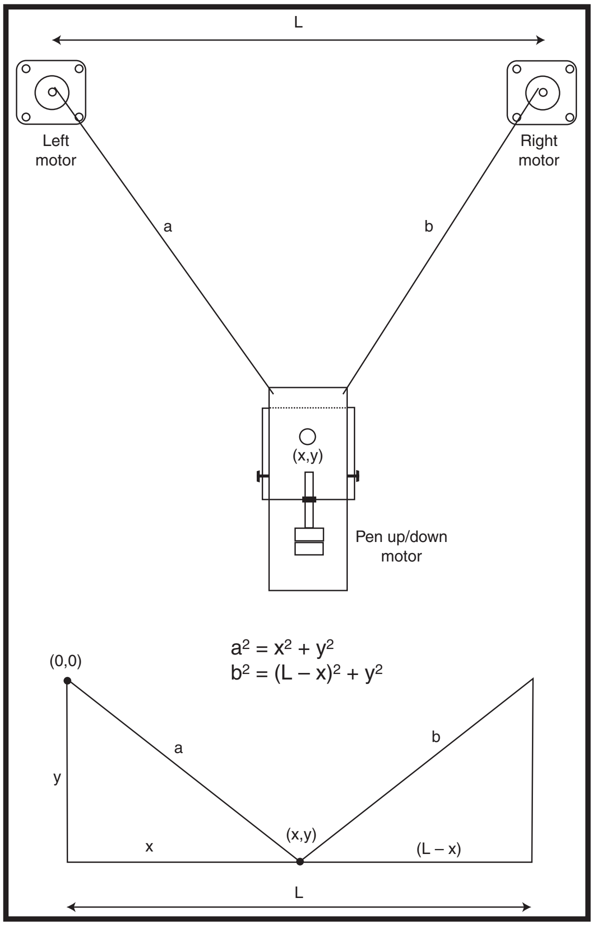


Figure 1—This is the system model and its associated mathematical model. Several simplifying assumptions are made here.

MECHANICAL SYSTEM DESIGN

The painting machine consists of two stepper motors (I had a couple of 1-A/5-V unipolar motors on my desk for several years) located on the top left and on the top right of the drawing area (see Figure 1). You can change the horizontal distance between them to accommodate a wider canvas. The unit holding the pen hangs from the motors' pulleys to form a V shape. Depending on the length of ribbon each stepper motor is releasing, the location of the pen holder changes. Gravity pulls the pen holder downward, so motors aren't required to pull down the pen holder.

I was naive enough to think that I could use just some string from the stepper motors to the pen holder, but after a quick test, it became apparent that any sliding on the pulleys ruined the result. So again, I followed what the Hektor makers did and used toothed belts (like your car's timing belt but open-ended) and toothed pulleys on the stepper motors. I used an MXL-sized toothed belt (5.08 mm) and 10 teeth pulleys (a revolution of the stepper motor is 50.8 mm in belt length).

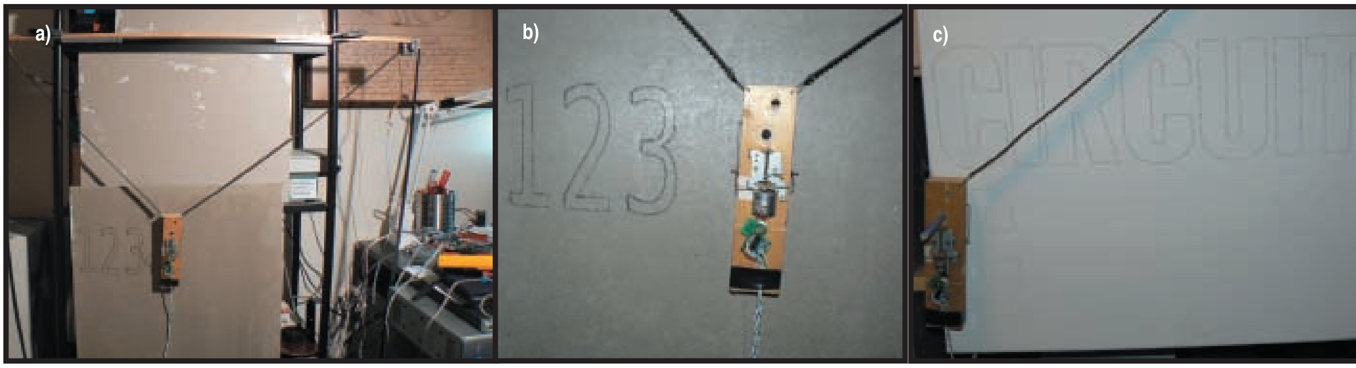


Photo 1a—Check out the system in action. **b—**This is a close-up shot of the output of one of the sample files. **c—**Here the output is "CIRCUIT CELLAR."

A word of warning: you do not want to machine a steel pulley with your home drill, trust me. If you are buying the hardware, try to match the stepper motor axle diameter to the pulley bore.

A couple of bearings press the belts against the pulleys to keep the teeth inside the pulley slots. This ensures system accuracy, which is important because you are going to work on an open-loop system (no position feedback).

The pen holder mechanism I used involves a third stepper motor. I obtained it from an old floppy disk drive in my employer's storage room. I noticed it was a bipolar motor (only four wires), so I sawed out part of the original PCB of the drive so I could keep the controller IC. (Fortunately, it was still in working condition.)

Wires were soldered to connect the controller board to the pen holder. It isn't very elegant, but it's effective.

CONTROLLER BOARD

At first, I planned to use a PC to control this device—after all, a user would draw designs on a computer—but I did not have any suitable hardware to use for this project. Fortunately, after searching the Internet, I discovered a USB-based stepper motor controller from an Australian company. It features a Cypress Semiconductor microcontroller and a couple of ULN2803 line drivers (see Photo 2). The board controls three unipolar stepper motors up to 1 A/30 V and includes some sample software.

While working with the board, I realized that the board had 12 open collector outputs, which wasn't exactly what I would call a stepper motor controller, but it worked well with my two unipolar motors. The software included an ActiveX component that I could include in a Visual Basic project. Sample code was provided. I hadn't used Microsoft Visual Basic for quite a while, but after downloading the free Express Edition, I was up and running in no time.

I tried to use a solenoid to lift the pen up and down, but given my lack of mechanical skills, the system wasn't operating properly. So, I switched to a more elaborate mechanism, where the rotation of the motor translates into a longitudinal motion that moves a metal lever so the pen touches (or doesn't) the canvas's surface (see Photo 3).

The pen's up/down system uses a floppy's bipolar motor. But outputs of the controller were not suited for a bipolar motor. I kept a small piece of the PCB from the original floppy disk driver circuit (see Photo 4). This made the bipolar motor easy to interface using the unipolar outputs intended for a third motor. That was when I realized that full control of each output line (as opposed to a real stepper motor controller) proved quite useful here.

The bipolar controller used on the floppy drive was Sanyo's LB1656, which used a couple of logic signals (D1 and D2) to control the polarity of each one of the stepper motor coils (see Figure 2). A third logic level enabled the selection between Hold and Seek actions. Seek sends 12 V to the coils, while Hold is only 5 V (the higher the voltage, the more torque). I set the pin to Seek when I want the motor to move. I set it to Hold when I want it to stay still.

MATH

If you are familiar with the Pythagorean theorem (Gougu's theorem for Chinese readers), you can understand how the system works. If you aren't, trust me, it works.

The two motors operating the pulleys control the lengths of the two belts the pen holder hangs from (see Figure 2). The two motors are located at a particular distance (L) from each other. The left triangle is defined by the left motor, the pen holder and the intersection from the horizontal line from the pen holder, and the vertical line from the left motor. The hypotenuse of such a triangle is the square root of the sum of the squares of both legs (x and y).

The other triangle is similar, but it is formed with the left motor and with one leg being y, the same as the former, and the other being L-x. From the two triangles, it is easy to calculate the respective hypotenuses to locate the pen holder at the desired XY coordinate.

The stepper motors I am using are

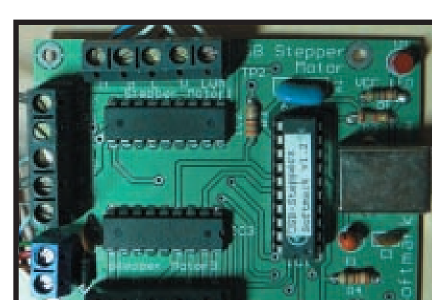


Photo 2—This is the USB stepper motor controller that is used in the project.

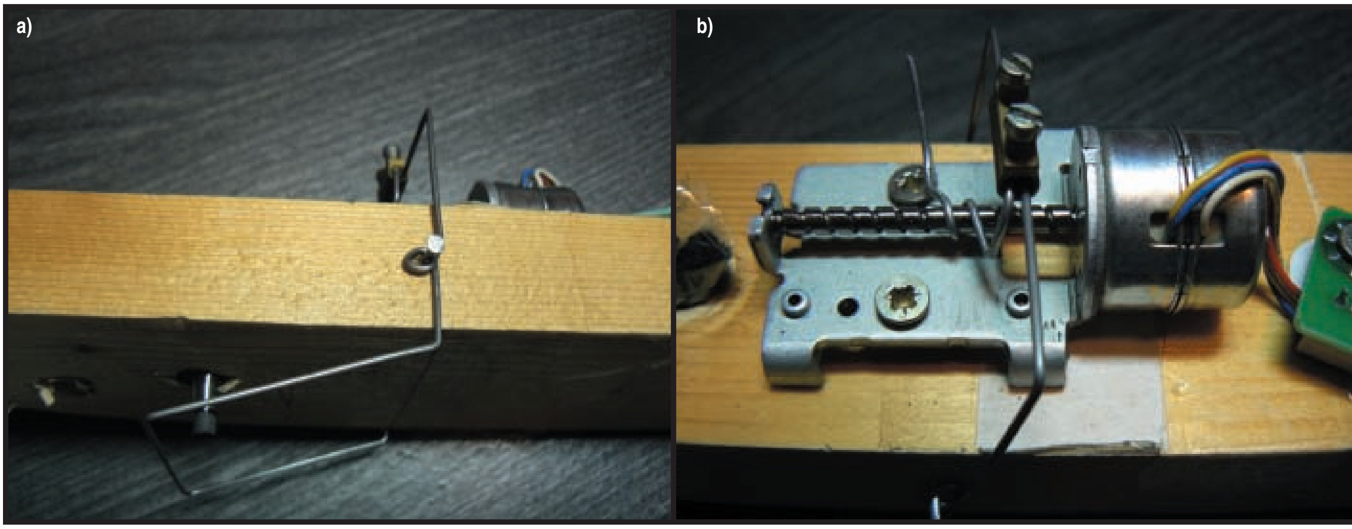


Photo 3a—This lever enables the pen to touch the canvas. **b—**This is the motor that raises the pen lever.

200 steps per turn. Because of the pulleys I use (10 teeth per turn) and the tooth distance 5.08 mm (or 0.2″), the spatial resolution is 10 × 5.08 mm/200 steps or 0.254 mm/step (or 0.01″ per step). So, given the desired Cartesian coordinate, it is easy to calculate the corresponding lengths for the left and right belts.

SOFTWARE

You know how to build a painting machine and you understand the math to tell the stepper motors what to do, but there are two more problems to resolve. The first one is easy. You need a way to establish the initial pen holder position. A home position has to be defined accurately because all of the belt's movements are relative to the previous location. If that is not correct, then you will get a pretty distorted drawing (which may be still usable if you are into the abstract art scene).

Like any other printer, you might need to conduct an initial routine to set the pen holder to the home location; however, you will need input lines to detect that the pen holder has reached a certain location. Unfortunately, my stepper motor controller does not have any input, so I manually locate the pen holder at a known location before starting any drawing.

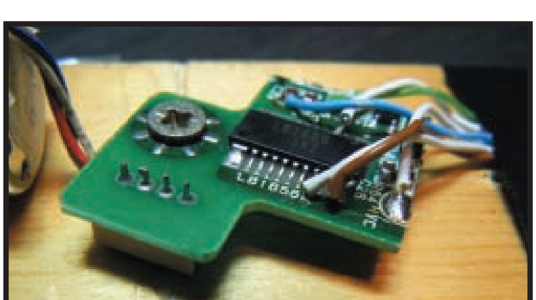


Photo 4—I saved the PCB from an old floppy drive with the bipolar stepper motor driver circuit.

The second problem is obtaining the sequence of commands to draw the lines you want the system to paint. I am familiar with vector drawing programs and the scalable vector graphics (SVG) file format. It happens to be an XML-based format, so you can easily look at it with any text editor program. It is also easy to parse.

I opened a new file on Inkscape software. I used a tablet to draw several lines as a test file. I then looked at the resulting lines and realized that most of them were Bezier lines. I mean curves. While I could write some code to turn that into a sequence of XY coordinates, I thought there had to be a better way. Fortunately, I discovered that Inkscape software has an effect called "Flatten Bezier," which turns these curves into a sequence of straight lines. Drawing a straight line in software is quite easy. Just use Bresenham's algorithm.

DRAWING AN IMAGE FILE

Your machine's readable SVG file format stores each path (processed as above) as a sequence of dots connected by line segments (also called multi-line). Before drawing a path, the pen holder has to be at the initial dot location. Next, the pen is set down and tracing starts. From the first dot towards the second dot of the path, the process repeats until the last dot of the path is reached and the pen is set up. The aforementioned sequence is repeated for each path of the drawing.

There is something counterintuitive about the dynamics of drawing a line in this system. Sometimes a motor rotates clockwise and then changes to counterclockwise while drawing the same straight line. This happens because the relationship between the XY coordinates and the corresponding belt lengths is not linear. From this observation, it is apparent that a stepper motor controller that you instruct with the final location and the desired acceleration shape will not help you here. To get a straight line, both motors have to move little by little and not directly to the final location.

A great and pleasant surprise was that Visual Basic includes support for regular expressions, which is a blessing when you are parsing text lines. (To see how this is done, refer to the code files posted on the *Circuit Cellar* FTP site.)

POSSIBLE IMPROVEMENTS

Getting the system working at an acceptable level took a lot of energy, so I left out some details. One

involves trying to reduce the total distance covered by the pen holder for a given drawing. This problem is also known as the travelling salesman problem.

If you do as I did, your image file's paths are stored in the file in a way that might be far from optimum when using a mechanical system like this one. However, this is not significant if you are drawing on a computer screen. The lack of this optimization means that the pen holder might be moving up and down all the time instead of concentrating on the paths on top of the image before starting to draw those on the bottom. This will translate into a much longer (but possibly entertaining) drawing time.

I contacted Professor Michael Trick, an expert on Operations Research at Carnegie Mellon University, who kindly provided me with a simple and elegant solution. A travelling salesman problem is built using all the endpoints of all the paths on the drawing. The cost between the endpoints of the same path is set to a negative value. For any other cost between endpoints, the Euclidean distance is used. The problem can be solved using "Concorde" software that's freely available from Georgia Tech University.

A second improvement I have in mind is to monitor the pencil. I want the system to warn me when the tip is too dull for drawing.

Although the hanging belts are modeled as straight lines, they are not. In fact, for those of you who are curious enough, the two belts form segments of a curve called a catenary. A catenary is the shape a chain forms when it is hanging from two supports under its own weight. While the error of my simplified mathematical model is acceptable for artistic renderings, it may be a bit of a problem for users who require better accuracy.

DESIGN AND DRAW

Despite the challenge, my vertical plotting machine was fun to build. In addition to its entertainment value, it is also pretty useful.

Please note that although your math might tell you that you can reach the top of the drawing area, you should keep a safe distance from the top, left, and right margins. This will ensure good accuracy when drawing. If you need to draw a wider picture, simply increase the distance between the two motors (or position your canvas in Portrait mode and rotate your image 90°).

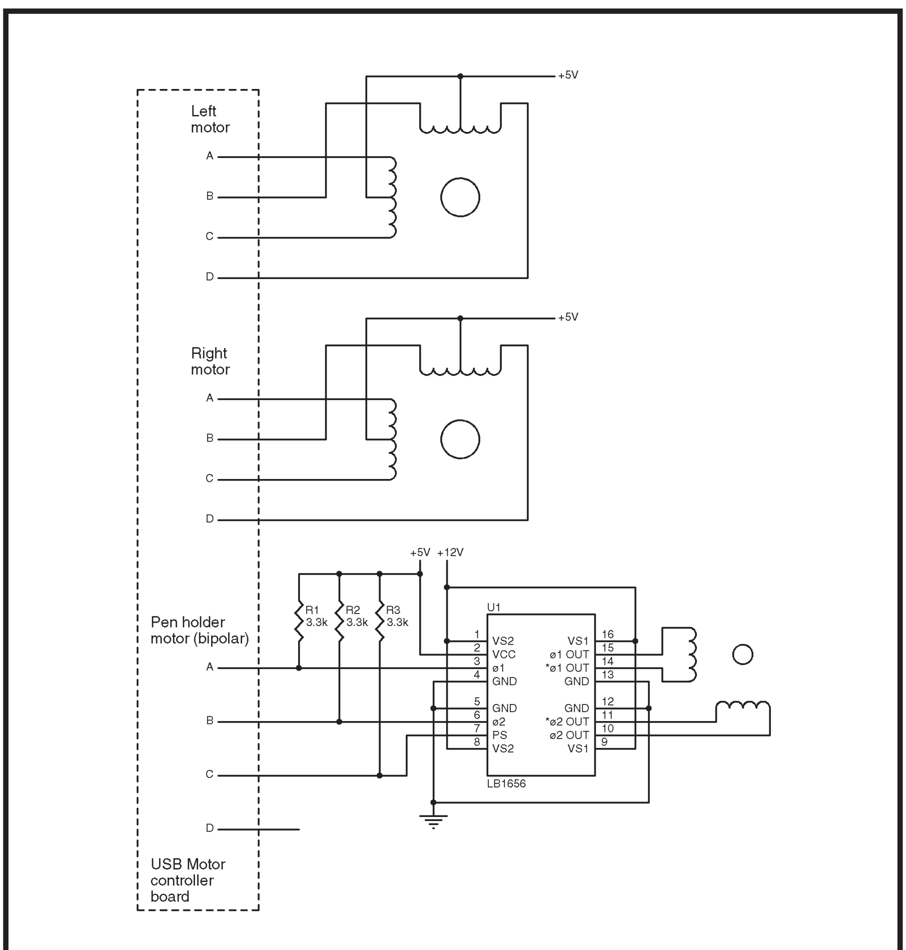


Figure 2—This is the wiring schematic diagram. Both unipolar stepper motors are shown. I also used a recycled bipolar stepper motor with a small hack.

Miguel Sanchez (misan@disca.upv.es) holds a B.S., an M.S., and a PhD in Computer Science. He has taught computer networks courses at the Technical University of Valencia, Spain, since 1989. Miguel's interest in electronics and microprocessors sparked his career in computer science, but his solder is always at hand. His research is currently focused on vehicle energy efficiency. Miguel also works as a consultant. A video of this project is posted at www.youtube.com/watch?v=VmB14M78CWU.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2008/212.

RESOURCE

Hektor project, http://hektor.ch.

SOURCES

Visual Basic 2008 Express edition
Microsoft Corp.
http://msdn.microsoft.com/vstudio/express/vb/

LB1656 Stepper motor controller
Sanyo Electric
www.sanyo.com

USB Stepper motor controller
Softmark
www.ar.com.au/~softmark

FEATURE ARTICLE by Michael Ghazi, Stefan Kaban, Scott Morken, Carl Philippsen, & Kyle Wong

Flight Control

A Communications System for an RC Helicopter

Whether you are looking to control an RC helicopter or a full-sized car, this system will work for you. By facilitating communication between a base station and a vehicle, the system enables an operator to process commands and receive data.

Anyone who lives near a model airplane club can attest to two things: the annoying buzzing of nitro engines and the amazing abilities of some of the scale models. Along with planes, model helicopters are sometimes encountered at these clubs. They are more maneuverable than their larger brethren, capable of performing aerobatic maneuvers and inverted flight. A characteristic that scale-model and full-sized helicopters share is their difficulty to pilot, especially for inexperienced technology students. So, when it came time to develop our final project in our technology program at Camosun College, we chose to work on an autonomous, self-balancing model helicopter.

An important part of this project involved developing a communications system for sending commands to the helicopter and receiving status information in return. Using simple, readily available hardware and extensive software, we successfully implemented the communications system to keep the base station connected to the helicopter.

In this article, we'll describe our communications system. It comprises firmware and a communications protocol.

APPLICATIONS

Our communications system was originally designed to control an RC helicopter. The system uses software, firmware, and a communications protocol to send commands to the helicopter and receive information from it. The command packets and data packets are transmitted between a base station connected to a laptop computer and a receiving module on the helicopter. Sensor data and user commands control an array of servo motors. With the right modification of servos and sensors, this concept for control could easily be implemented to aid in the control of other types of RC or full-scale vehicles, including boats, planes, or cars.

Who can use a system like ours? A search and rescue team could send out multiple autonomous helicopters equipped with cameras. The smaller-scale autonomous helicopters would have a better chance of finding people in distress than a single full-scale helicopter. Also, the smaller-scale autonomous helicopter would come with a smaller price tag than the full-scale version. But the applications for this communications system are not limited to search and rescue. This system could be used to help remove human involvement with virtually any vehicle's operation. You can use the system to control vehicles for cinematography, aerial mapping, law enforcement, surveillance, inspection, and more.

SYSTEM OVERVIEW

The communications system is designed to transfer data between the base station and the target device. Our target device is an electric RC helicopter. The communications system consists of PC software at the base station, hardware on the base station, hardware and firmware on the helicopter, and a communications protocol to tie everything together.

The PC software consists of three subsections: the communications class library, the indicator user control library, and the Google Maps user control (see Photo 1). The communications library links the GUI to the helicopter. It is built around a serial port connection and it provides a simple interface for

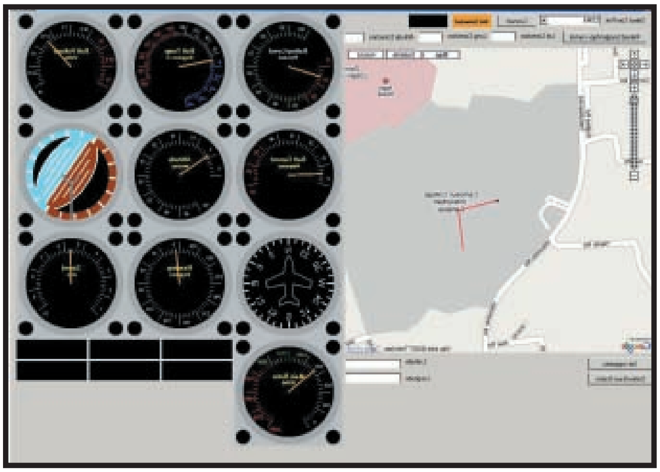


Photo 1—This is the graphical user interface for our system. The customizable indicators enable us to keep track of any data received by the base station.

the GUI coder. The indicators library shows the user telemetry data from the helicopter. It can display customizable dials, a compass control, an artificial horizon, and indicator lights. The overall indicator design mimics a helicopter or airplane instrument panel. The Google Maps user control defines way points for the helicopter and shows its position.

For base station hardware, we used an antenna, a USB-to-Serial module, and a transceiver module mounted to a PCB. The hardware on the helicopter for the communications system consists of an antenna, a transceiver module, and a Microchip Technology dsPIC30F3011 microcontroller. The communications protocol is a rule set that governs the transmission of data between the base station and the helicopter.

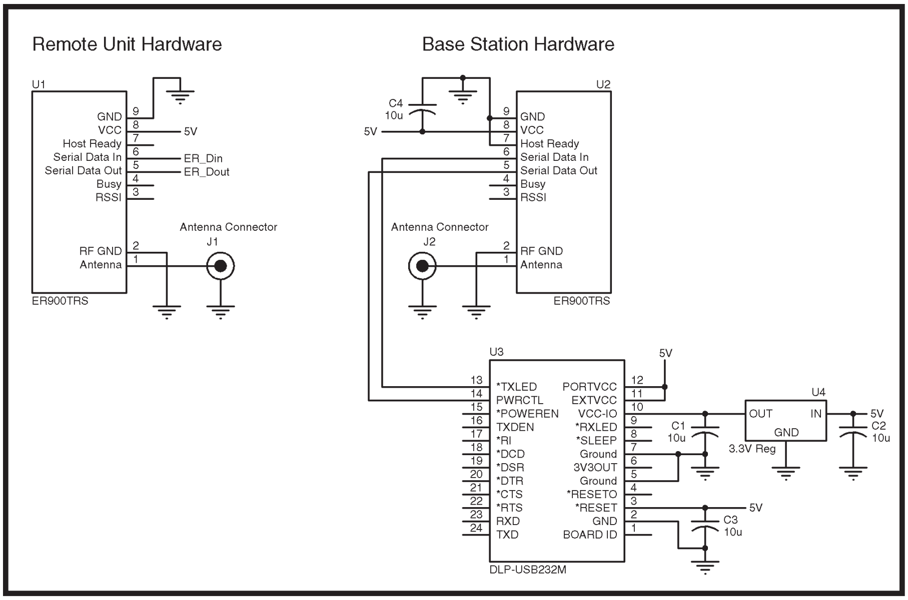


Figure 1—This is a schematic of the base station and remote unit hardware.

HARDWARE

We knew from the start that the most difficult aspect of this project would be the software. For this reason, we used simple pieces of proven hardware to get the job done so we could have more time to focus on the software. The communications system hardware on the helicopter is virtually the same as the hardware in the base station.

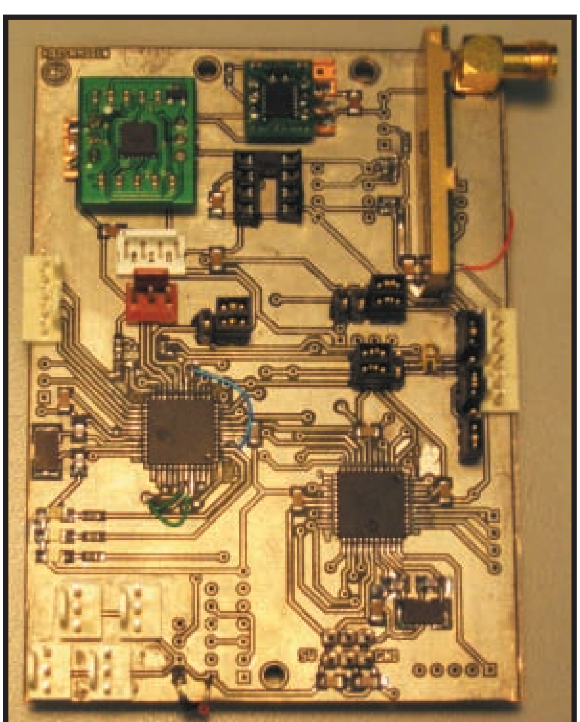


Photo 2—This is the main control board of our helicopter. Visible at the top right of the photo are the radio module and the antenna connector.

We used a pair of Low Power Radio Solutions ER900 TRS 900-MHz transceiver modules to send data between the base station and the helicopter (see Figure 1). We chose the modules for their simple serial interface, their operation in an unlicensed band, and their ability to take care of error handling. The transceiver modules are mounted to PCBs, which we designed and manufactured at Camosun College (see Photo 2). The base station board, which holds one of the transceiver modules, also holds a USB-to-serial converter. The USB-to-serial converter is the link between the transceiver module and the software on the PC.

The helicopter radio module is coupled to a 4″ omni-directional whip antenna and interfaced to the USART of the onboard microcontroller. This antenna works well on the helicopter because it is small and fairly lightweight. For a base station antenna, we started with a second 4″ omni-directional whip antenna. The signal was somewhat limited using this antenna, so we chose to upgrade to a 12-dBm, 4′ long, nine-element Yagi-Uda antenna. This gave us increased range and better signal quality. Plus, we got it for free because one of our instructors purchased it on the condition that we give it to him (for one of his evil schemes) when we were done.

PERFORMANCE EVALUATION

We tested the communications system and determined the data rate was 517 bps. We calculated this with a measured time per byte of 17.4 ms, which is close to the datasheet's minimum specified value of 15.978 ms. The latency of the packets varied depending on packet length, with an average delay of 46.75 ms. We measured this delay as the time from when a set of outgoing data is loaded into the radio's buffer to when an acknowledgement is received. Our latency measurement does not take into account the unpredictable delay caused by the non-real-time nature of Windows XP.

The message throughput can vary depending on which error rate is considered acceptable. We made two measurements. At 3.03 packets per second, we determined the error rate was 2.05%. At 4.08 packets per second, the error rate was 2.33%. The two packet rates were used to illustrate the trade-off between the packet throughput and the number of errors that occur. As packet throughput increases, the error rate also increases.

PROTOCOL

The protocol we developed is a set of rules that enables us to maintain

Header		Length	Data			Checksum		Footer	
0xA5	0x5A	0x03	0x54	0x45	0x32	0x00	0xCE	0xCC	0x33
—	—	—	Family	Command	Payload	Sum of data and length		—	—

Table 1—This is an example of a packet sent between the base station and the helicopter. This packet, when received and decoded by the helicopter, will set the speed of the engine.

reliable communication between the base station and the helicopter. Work on the communications protocol began even before the official start date of our project. This was a large task, which we needed to complete quickly so we could use it to test other aspects of the helicopter's operation. The full code communications protocol is available on the *Circuit Cellar* FTP site.

The communications protocol is fairly simple in the sense that we did not incorporate any error checking. This was possible because the radio modules we used have some good error-checking routines of their own. There are several bytes within the protocol that have specific tasks. The header bytes are used to "wake-up" the receiver so it can begin recording and decoding the incoming transmission. The length byte tells the receiving station how many bytes are in the data portion of the packet and includes the command group, command, and payload portions. The command group byte tells the receiving station (either the helicopter or the base station) what type of command is being sent. For example, it might be a testing/tuning command, in which case the command group byte would be 0x54. We included a command group to make decoding simpler and better documented. Another programmer looking at the state machine can easily go to the group in question and have fewer commands to sift through.

We use command bytes that make sense. For example, the hexadecimal number 0x50 is an ASCII character "P." This is the character used in the command to change the pitch servo pulse width ("P" for pitch). However, "P" is also the command byte for the base station to request a preflight packet ("P" for preflight). The command group byte differentiates the two. The command byte then tells the receiving station exactly what command is sent. In Table 1, 0x45 indicates an engine speed change. (Hexadecimal number 0x45 is an ASCII character "E," for engine speed.) The checksum is used to verify that the data received is the data that the transmitter intended to send. It is defined as the two least significant bytes of the sum of all the bytes in the length and data portions of the packet. The footer simply indicates the end of a transmission.

In our master/slave system, the base station is the master and the helicopter is the slave. This has a number of implications. The base station initiates all communications, and the helicopter doesn't transmit unsolicited data. The only exceptions are error messages, because the base station can't know when an onboard error will occur. To ensure that all transmissions from the base station are received, each one is acknowledged by the helicopter in one of two ways. Command packets are acknowledged by the helicopter by transmitting the entire packet, plus an ACK byte (0x06, which comes after the length byte). Information requests are acknowledged by simply sending the information requested. If an error is detected in the packet, the packet is ignored and no ACK is sent. It is the master's responsibility to retransmit any commands or information requests that are not acknowledged.

Table 2 is an overview of the complete communications protocol. For a detailed description of all packets and payloads, refer to the Communications Protocol document posted on the *Circuit Cellar* FTP site.

Group description	Group byte	Command	Command description
Testing/tuning	0x54	0x45	Engine speed adjust
	0x54	0x50	Pitch servo adjust
	0x54	0x52	Roll servo adjust
	0x54	0x43	Collective servo adjust
	0x54	0x51	Anti-torque servo adjust
	0x54	0x66	Set operations mode
	0x54	0xDD	General-purpose data dump
Flight operations	0x46	0x45	Engage engine
	0x46	0x48	Hover
	0x46	0x43	GPS Correction factor
	0x46	0x47	Go to GPS coordinates
	0x46	0x52	Return to base
	0x46	0x50	Request pre-flight packet
	0x46	0x4D	Discreet movement
	0x46	0x49	Request for information
Telemetry data	0x74	0x4C	Location
	0x74	0x48	Heading/speed/altitude
	0x74	0x5A	Attitude
	0x74	0x42	Battery status
	0x74	0x45	Error report
	0x74	0x50	Preflight packet
	0x74	0x52	Rotor RPM

Table 2—This is a listing of the communications protocol used in our system. The protocol is flexible and it can expand to up to 65,535 commands, which would be broken into 256 groups. As an added bonus, the code is easily modified to accommodate the extra commands.

FIRMWARE

The data sent via our protocol has to be decoded and used once it is received by the helicopter. This task is performed by our firmware program. The firmware handling our communications is loaded onto a dsPIC30F3011 on the helicopter. We chose this microcontroller because it has powerful processing abilities, which we use in our helicopter for control calculations, and because we happened to have a stash available to us at school.

The firmware, which is essentially an onboard packet parser, comprises two separate state machines. The first state machine checks the packet header and data length, copies the data into a buffer, verifies that the checksum is correct, and checks if the footer is valid. If everything is fine, it sets a valid data flag. This starts up the second state machine, which parses the valid data packet. It checks the first byte to determine which command "family" the command belongs to and

checks the actual command. It then determines which action is appropriate to take based on the command. Table 1 shows a complete example of an engine speed packet.

SOFTWARE

The PC software is designed to keep you in control of the helicopter and informed of its status. It has three main components: the communications class library, the indicators user control library, and the Google Maps user control. All parts of this software are coded in C# using the .NET framework and the Microsoft Visual Studio 2005 IDE.

The communications library is used to link the GUI to the helicopter. It is built around a serial port connection. The library provides a simple interface for a GUI coder to use the communications protocol. It enables all packets to be sent by a simple function call and each received packet raises an event for the GUI to handle. All other functionality, such as error checking and packet parsing, is dealt with internally by the communications class library. The library is centered on a .NET serial port object and a high-speed timer. Each time the timer event is raised, the library checks the serial port buffer to see if its contents are a valid packet or if there is a transmission error. The serial port object raises an event when bytes are received, but we found the event was not raised reliably on every byte. Additionally, using the event forced us to pass the event data across threads. This is because the serial port's events are on a separate thread from the GUI. Because the PC software initiates most communications with the helicopter, it is essential to receive the helicopter's acknowledge packet to verify that the desired data has gone out. For this reason, each time a packet is sent, a timer is started. If the timer expires before the expected response is received, a timeout event is sent to the GUI.

The indicators library is used to show the user telemetry data from the helicopter. It is highly customizable: we use it to create the indicators from a real aircraft control panel (i.e., a compass, an artificial horizon, and indicator lights). All user controls are scalable in size, enabling them to be resized on a form size change or other events. Dial indicators can be customized easily for many different applications. The numeric range, labels, number of ticks, color ranges, and other aspects can be modified to suit any application. The left-most dial in Photo 1 is an example of a temperature dial that has been set up with a –40° to 100°C range and warning bands for "under" and "over" temperature.

We drew all of the indicators using the graphics device interface (GDI). We started by making the gauge class, which inherits the basic Windows user control and implements all features that are common to center dial, artificial horizon, and compass classes. In turn, each of the classes inherits the gauge class. An important thing we learned was the value of using double buffering to prevent flickering of the things drawn using GDI.[1]

The Google Maps user control shows the position of the helicopter with GPS coordinates received via the communications protocol. It is also used to define way points for the helicopter. The user control requires an Internet connection. It is based on a .NET web browser control that loads Power Map, a JavaScript page containing interface functions to use with Google Maps. The JavaScript on this page forms a wrapper around the Google Maps API. It is what the map control interfaces to. Each time a GPS packet is received from the helicopter, we convert the packet's contents from the standard NMEA string format (www.geoaps.com/NMEA.htm) to floating-point degrees. The map can be easily centered on the coordinates or a marker can be placed at this point. At startup, the Google Maps user control loads the Power Maps JavaScript. If an Internet connection is present, the Google Maps interface will load and a map will be shown. Each JavaScript function in Power Map is called by the method HtmlDocument.InvokeScript method.[2]

IMPROVEMENTS

If you have the time and money, we would recommend adding error (parity and redundancy) checking to the communications protocol. We were limited on time, so we skipped this aspect of the project because our radio modules had error checking built in. Adding

error checking to the communications protocol would enable the protocol to be used on any radio module. Users of the communications system would not have to worry about finding a radio module with error checking built in. They would be able to find any radio module and plug it in.

Another important improvement would be to modify the protocol to allow for multiple slave devices, enabling the multiple helicopter scenario described in the introduction. This would require changing the protocol packets and firmware to include an address byte (or bytes).

The Google Maps control is coded specifically for our helicopter project, so another recommended improvement would be to make the Google Maps control more general-purpose and extend its functionality. There are several functions written in the JavaScript that are not yet implemented in the .NET control. One example is importing way points from an XML file.

We also have some recommendations to improve the indicators class. Currently, it is made to simulate an aircraft control panel. It would be beneficial to make it more general-purpose as far as how it looks. Also, adding more controls, such as thermometers, seven-segment displays, or redline range events, would add to the functionality. ♟

Michael Ghazi's (michael@ghazi.ca) six years as a naval communicator in the Canadian Forces prompted him to undertake the task of designing the RF system for the helicopter. After completing his studies at Camosun College, he will be relocating to the greater Toronto area to find work in his field.

Stefan Kaban's (snkaban@gmail.com) interest in helicopters was piqued during a co-op job with the B.C. Ministry of Forests, where he was regularly exposed to helicopter operations. He plans to complete his degree in Electrical Engineering at the University of Victoria.

Scott Morken (scottm361@gmail.com) is a graduate of the Computer Engineering Technology program at Camosun College. He discovered his new favorite programming language while working for two co-op terms as a C# developer. Scott has also worked on many other hobby software projects. Updated versions of some of the project software are available on his web site (www.red79.net/Projects.html).

Carl Philippsen (carlphilippsen@hotmail.com) is a graduate of the Electronics Technician Common Core program at North Island College and the Electronics Engineering Technology program at Camosun College. He is interested in a career in biomedical engineering and plans to further his education at the University of Victoria.

Kyle Wong (kindawong@gmail.com) worked as an electronics technologist for several years. He plans to attend the University of Victoria and enroll in the Electronics Engineering degree program.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2008/212.

REFERENCES

[1] G. Schmidt, "Don't Flicker! Double Buffer!," The Code Project, 2007, www.codeproject.com/csharp/DoubleBuffering.asp.

[2] Microsoft Developer Network, "HtmlDocument.InvokeScript Method (String)," http://msdn2.microsoft.com/en-us/library/be9zzz62(VS.80).aspx.

RESOURCE

R. Scammell, Power Map, http://hobbiton.thisside.net/advmap.html.

SOURCES

ER900 TRS 900-MHz Transceiver modules
Low Power Radio Solutions
www.lprs.co.uk

dsPIC30F3011 Microcontroller
Microchip Technology, Inc.
www.microchip.com

Visual Studio 2005 IDE
Microsoft Corp.
www.microsoft.com

by Chris Coulston

Do-It-Yourself Motion-Controlled Gaming (Part 1)

Get Started With a Simple Application

The Nintendo Wii gaming system features a motion-sensing remote that translates hand movements and gestures into actual moves in its games. In this series of articles, Chris describes how you can build your own motion-sensing controller that can interact with a variety of graphics programs running on a PC.

One of the joys of teaching an embedded systems class is exploring new technologies and sharing the discoveries with students. To give my students a chance to apply what they have learned, they choose their own final projects. Last year, one final project team used a Microchip Technology PIC18F452 to read acceleration data from an Analog Devices ADXL220 and send it to a PC. They created an OpenGL application on the PC to use the ADXL values to change the tilt of a rendered 3-D maze containing a ball. The user guided the ball through the maze towards a goal. The students voted this the top project of the year. Not one to shut the door on opportunity, I wondered how I could incorporate the project into class assignments. Not long after that, Nintendo came out with the Wii, a gaming system based around a motion-sensing controller. Hence, the "Do-It-Yourself Wii" was born.

The Nintendo Wii consists of a console that is plugged into a TV, a controller, and a sensor bar. The user controls the on-screen action with a controller containing a variety of buttons, a three-axis accelerometer, and an optical sensor. The acceleration information provided by the accelerometer enables the user to interact with console games using physical gestures like punching in a boxing game or rolling a bowling ball in a bowling game. The sensor bar's pair of IR LEDs are detected by the controller's optical sensor, enabling the controller to be used as a pointer device.

Photo 1—This is the completed DIY Wii. Don't let those tiny SMT devices intimidate you. All of my 30 students got their boards to work.

The goal was to have students in my class assemble their own DIY Wii controller and build an embedded application and PC applications. This would enable them to interact with graphics programs running on a PC using gestures in the same way that gamers use the Wii controller to interact with games running on the Wii console.

REQUIREMENTS

While the eventual goal of the project was to have students build a device that could communicate acceleration information to a PC, it also needed to be flexible enough to realize a variety of standard embedded systems projects. These projects would need to demonstrate concepts such as PWM, frequency modulation, parallel and serial communication, and switch debouncing, to name a few. Given the diverse backgrounds of the students, the system needed to be easy to assemble, the construction of the device could not require any sophisticated techniques, the system had to be relatively inexpensive, the software development tools had to be inexpensive so students could purchase them and work on their home PCs,

and finally the system had to be self-contained (not requiring special power supplies or other non-essential support hardware). The device that was created to meet the requirements is shown in Photo 1.

As you can see, the board comes equipped with lots of I/O devices to demonstrate embedded concepts. A combination of good silk screening on the PCB, detailed construction instructions, and a novel soldering technique resulted in a system that was straightforward to assemble. The system was reasonably inexpensive. The unit cost of the parts and PCB was $43. To keep costs down, I jettisoned the optical sensor, foregoing the ability to use the DIY Wii as a pointer. Students who wanted to work at home had to purchase their own $35 PICkit 2, a USB device that programs the microcontroller through a six-pin header. Software development took place in Microchip's free MPLab IDE using the free student edition of its C compiler. AAA batteries and some onboard boost converters were a perfect match for the DIY Wii's diminutive power requirements. Before discussing its capabilities, let's take a brief look at the hardware that makes up the DIY Wii and the motivation for these choices.

HARDWARE DESIGN

During the previous year in my embedded systems class, we used the PIC18F452, which in the interim year had evolved into the PIC18F4520. A quick check of the technical documents revealed some small differences in the CCP subsystem; otherwise, the parts were nearly identical. Thus, using the PIC18F4520 meant that I could keep a large portion of my lecture notes unchanged—that made the microcontroller choice obvious. The rest of the DIY Wii is built around the microcontroller to support its operation and enable the delivery of standard embedded systems labs (see Figure 1).

The center piece of the board is an Analog Devices ADXL330. Because of investigations by the folks at SparkFun Electronics, I know that this is the same accelerometer used in the Wii controller. My excitement about using this part was tempered by the knowledge that it came housed in a difficult-to-solder 16-LFCSP package. The bypass capacitors on the ADXL330's outputs were selected to give the device a bandwidth of 50 Hz, fast enough for most human movements. At the outset of this project, I knew that Microchip's ICD2 module was capable of programming the part, but at $160, it was not an option the students would be able to afford. At $35, I thought the PICkit 2 would be a great programmer, but I was not certain how it would perform in a punishing laboratory environment because

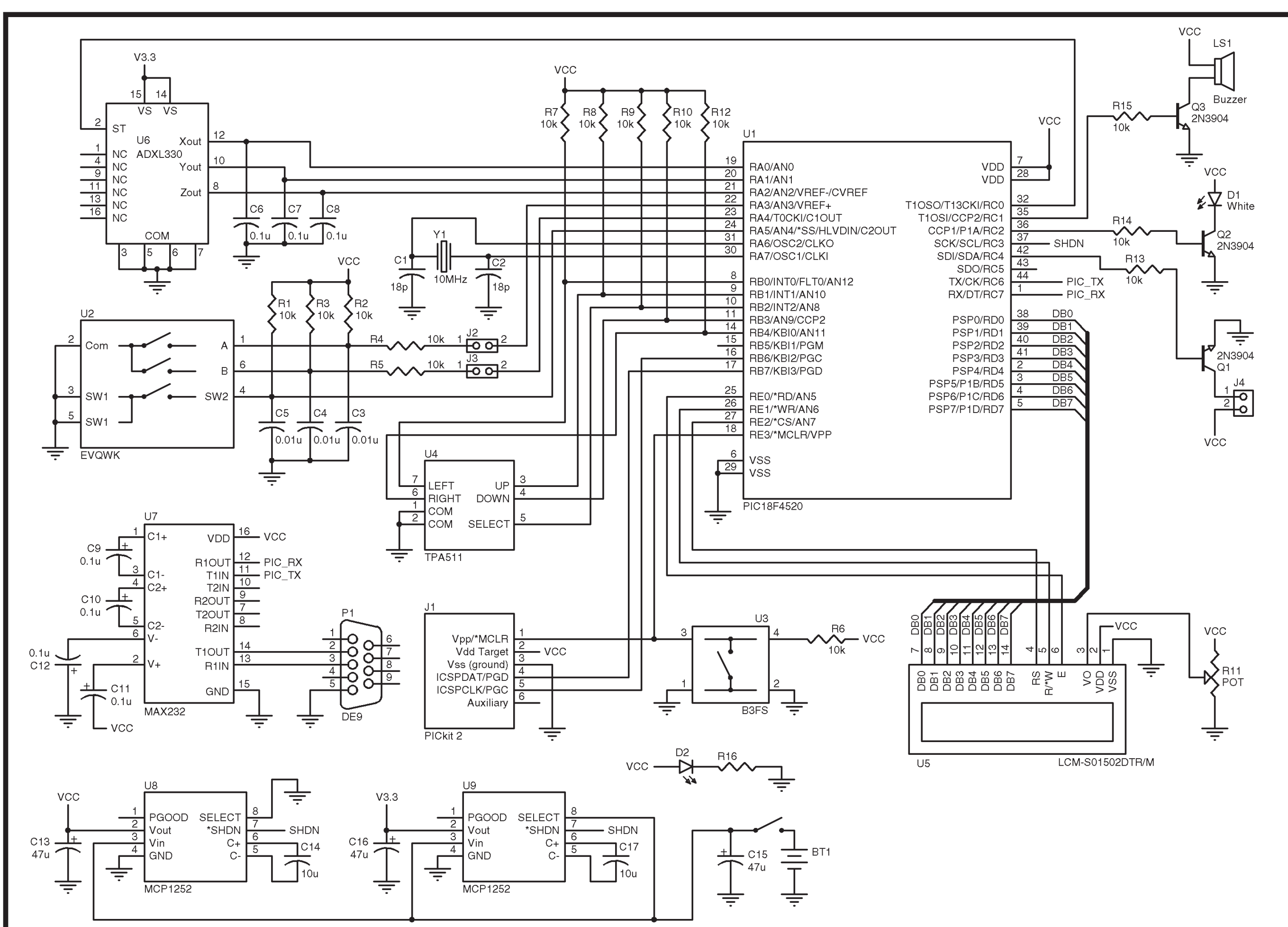


Figure 1—The schematic for the DIY Wii seems very busy, but those small SMT devices mean that the design has a small overall footprint.

I had never used one. Instead of risking the entire project on this choice, I put redundant programming headers on the board—an RJ-45 jack for the ICD2 and a six-pin header for the PICkit 2. As it turns out, the inexpensive PICkit 2 worked great and I never installed the RJ-45 jack for the ICD2. I wanted to put a Panasonic edge drive rotary encoder on the board, but at \$10, I decided that the \$1 ITT Industries four-way tactile switch with select would give the students enough control over user input. Looking at Photo 1, you can make out the empty space reserved for the Panasonic EVQWK edge drive jog encoders. 2N3904s were used to isolate the microcontroller from the high current requirements of the white LED and piezo speaker. An optional output driver was left open for future expansion. The Lumex LCM-S01602DTR/M LCD is an exceptional value for the money and it's very durable. The SMT 10-kΩ potentiometer was a disappointment; it proved to be easy to damage mechanically and thermally. It would have been a far better idea to fix the contrast of the LCD by putting a 100-Ω resistor between the V_O input on the LCD and V_{CC}. Two separate power supplies were installed because the LCD required 5 V and the ADXL330 required 2 to 3.6 V. The MCP1252 was a great part and it performed well. The 3.3-V boost converter could have been eliminated by noting that the anode of the green power-on-indication LED sits at 5 V and its forward-bias voltage drop is 2.1 V. Hence, the LED's cathode sits at 2.9 V, which is good enough for the ADXL330. Furthermore, the ADXL is an electron miser, consuming a scant 320 µA of current, hardly enough to affect the brightness of the LED. A standard MAX232-level converter and a female DB9 connector were used to communicate between the PC and the DIY Wii. My only real mistake in the PCB design was reversing the TX and RX lines between the DB9 and the level converter. This mistake provided me with the opportunity of introducing my students to "green wire" fixes.

ADXL330

The ADXL330 is a three-axis accelerometer capable of measuring forces up to ±3 G. It is available only in a 16-"pin" leadless chip carrier. Figure 2 shows the ADXL330 superimposed on top of its three acceleration-sensing axes. The white dot indicates the position of pin 1 on the physical package.

The acceleration along each of the sensing axes is converted into a voltage proportional to the acceleration. When the part is accelerated along an axis, the associated output voltage will increase. The output is ratiometric to the input acceleration and depends on the supply voltage. With a supply voltage of 3.3 V, the 0-G output is 1.7 V and each axis has a sensitivity of 330 mV/G. Thus, the ADXL shown in Figure 2 would have outputs of $V_X = 1.7_V$, $V_Y = 1.7_V$, and $V_Z = 2.03_V$.

There is only one configuration detail that must be addressed when incorporating the ADXL330 into a design: the selection of the bypass capacitors. The bypass capacitors determine the tradeoff between bandwidth and noise. The bandwidth determines the highest frequency accelerations that can be reported by the device. Clearly, having high bandwidth would be desirable. You could let the unused bandwidth go to waste. However, bandwidth is "purchased" at the expense of noise. Noise manifests itself in spurious variations at the voltage outputs. The ADXL330 technical documentation explains that "the ADXL330 noise has the characteristics of white Gaussian noise, which contributes equally at all frequencies and is described in terms of µg/√Hz..." Thus, it makes sense to ask questions about the probability of the noise exceeding some threshold. Clearly, low noise would be desirable because everyone likes precise measurements. Unfortunately, the device does not enable you to have both high bandwidth and low noise. Hence, you need to determine the minimum bandwidth in your application and the maximum tolerable noise. In the case of the DIY Wii, we were primarily focused on measuring human movement. For this reason, we chose a conservative 50-Hz bandwidth and tolerated the resulting noise.

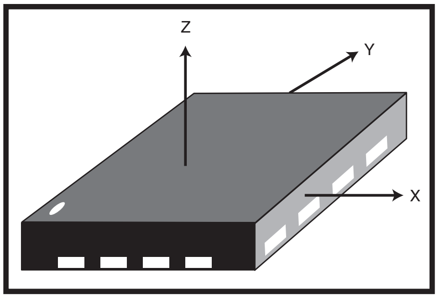


Figure 2—This is the ADXL330 with its axes of acceleration sensitivity.

SMD SOLDERING

If you are like me, then you have avoided SMD parts because you believe that you lack soldering skills, you don't know how to (or don't want to) build a custom PCB, or you are put off by the cost of expensive adapters for breadboards. Unfortunately, more and more cool devices, such as the ADXL330, are available only in surface-mount packaging. I figured that if we were going to get our toes wet, then we might as well jump in the deep end and go with surface-mount packaging whenever possible.

With the decision made to go with surface-mount packaging, I now faced the prospect of students without soldering experience assembling the DIY Wii. There are two general ways to solder parts to a PCB: hand soldering and reflow soldering. Hand soldering is very versatile. With a good soldering iron, the right solder, and a bunch of solder wick, virtually any surface-mount device (SMD) can successfully be attached to a PCB. One technique is to melt a little solder on one of the PCB pads and let it cool. Next, align the device on its PCB pads, hold it down, and heat up the pin over the soldered pad. This will cause the solder under the pin to melt. After the soldering iron is removed and the solder is allowed to cool, your part will be tacked down to the PCB. You can solder the remaining pins with abandon. This will undoubtedly create solder bridges, chunks of solder spanning consecutive pins. Don't worry about this. You can clean the bridges up with your solder braid, making a top-notch connection. Unfortunately, the success of hand soldering depends heavily on the skill and patience of the user.

The other soldering process, reflow

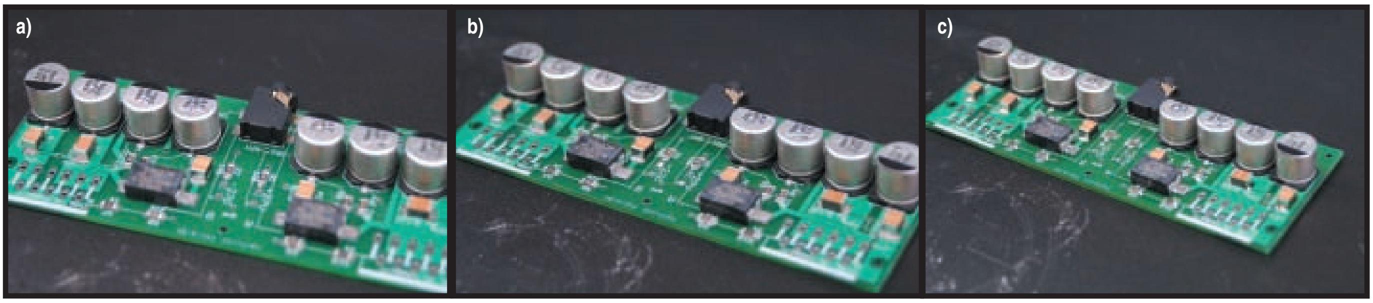


Photo 2a—Here is a pasty PCB waiting to be cooked on the reflow skillet. **b**—Five minutes into the process, the paste looks baked dry. **c**—Eight minutes into the process, the pads are looking shiny. This is reflow!

soldering, uses a viscous form of solder called solder paste. Reflow soldering has the advantage of requiring little skill beyond putting a dab of paste on the PCB pads and dropping parts onto the pasty pads. This assembly is then "cooked," turning the solder paste into solder. The cooking proceeds according to a thermal profile that can be broken down into four phases: preheating, soaking, reflow, and cooling.[1] During preheating, the assembly is gradually brought up to a temperature below the melting point of the solder. During the soaking phase, the assembly is held at a constant temperature below the melting point of the solder to ensure that all of the parts are at a uniform temperature. The temperature is rapidly increased during the reflow phase, causing the now dry solder paste to liquefy, attaching the parts to the PCB. During the cooling phase, the assembly is allowed to gradually cool to avoid thermal stress on the parts. The temperatures and times for each of the zones characterize a thermal profile. The technical documents for a popular solder paste like Kester's R276 suggest a thermal profile. If you are planning on building a million cell phones, you will need a reliable high-yield process that will justify the expense of a sophisticated reflow oven. However, a careful look at the Kester technical documents reveals a caveat. "The profile shown is simply a starting point recommendation. A wide variety of profiles would work effectively with R276 paste." The folks at SparkFun Electronics took the challenge to see just how tolerant solder paste is to changes in its profile. They came up with a low-cost, effective, and innovative technique called the reflow skillet.

The reflow skillet is nothing more than an electric griddle commonly used to cook on. I bought a Rival removable plate griddle for $30 from my local big-box store. It has a 10″ × 20″ cooking area (large enough for even the largest PCBs) and an easy-to-access temperature knob. I don't need to tell you that once you use the griddle to reflow a PCB with its associated heavy metals, it should never again be used to cook food, right?

PERFORMANCE

The biggest question before using the reflow skillet was whether it would really work. I also wondered about how the skillet's thermal profile would compare to the recommended profile for the solder paste. The answer to the first question could only be answered by reflowing a DIY Wii board. The answer to the second question required a trip to my local tool store for an infrared thermometer with a laser sight.

With everything at hand, I applied solder paste to all of the SMT pads on the DIY Wii PCB and populated the board by gently laying the parts onto the pasty pads. My nerves were on edge when I popped my first pasty PCB onto the reflow skillet and cranked the temperature knob all the way to 400°. Photo 2a shows a PCB (not the DIY Wii) at the start of the reflow process. If you look carefully, you can see that the solder paste is still a little bumpy.

After 3 minutes, the PCB was starting to warm nicely and the solder paste at this point was evened out across the pads as if the heat were somehow melting it. After 5 minutes, the solder paste was dried to a baked consistency (see Photo 2b). At about the same time, little whiffs of smoke started curling out from under the PCB. While this initially freaked me out, I have since come to the conclusion that this is normal and does not damage the bottom-side solder mask.

After about 8 minutes, some of the solder paste in the hotter regions or areas with little thermal mass started to reflow. It took 10 minutes for the solder

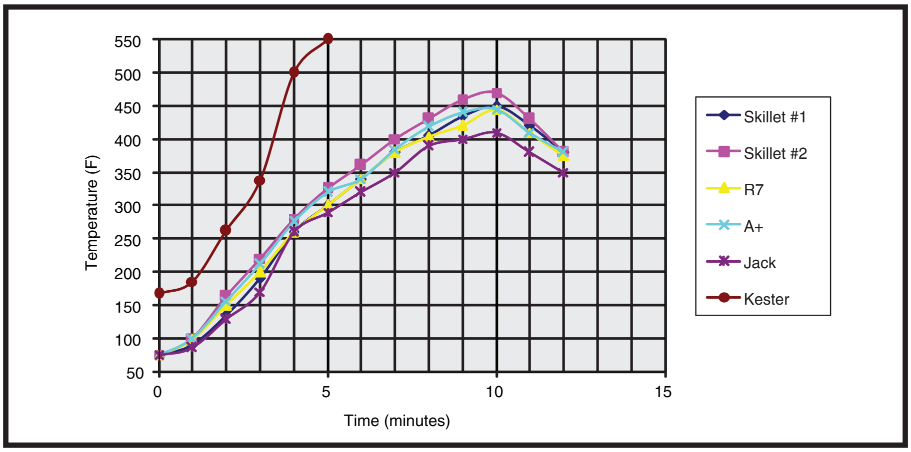
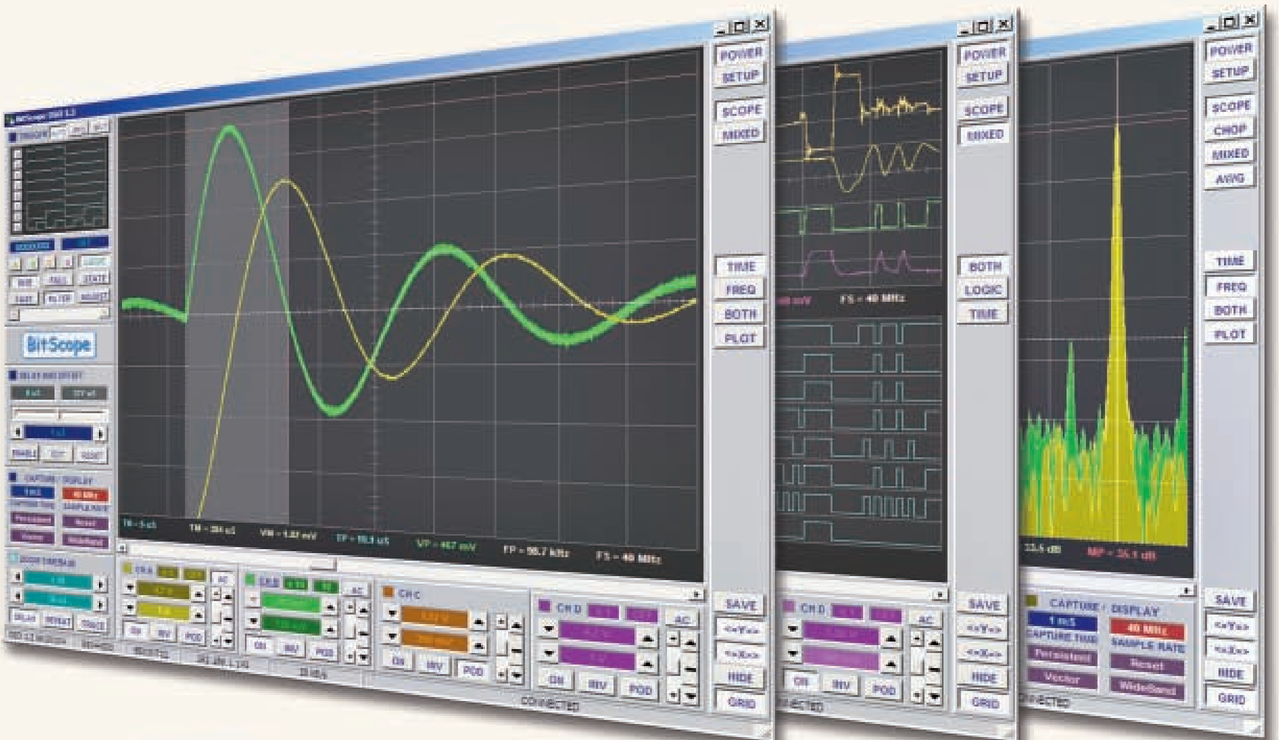
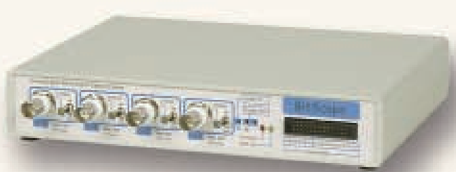


Figure 3—Here is the reflow profile suggested by the Kester documentation. The profile is generated by the Rival removable plate griddle at a variety of points on the skillet and PCB.

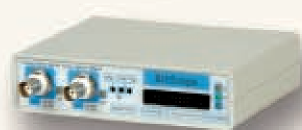
BitScope PC Oscilloscopes & Analyzers

DSO Test Instrument Software for BitScope Mixed Signal Oscilloscopes





4 Channel BitScope



2 Channel BitScope

Pocket Analyzer

Digital Storage Oscilloscope
✓ **Up to 4 analog channels using industry standard probes or POD connected analog inputs.**

Mixed Signal Oscilloscope
✓ **Capture and display up to 4 analog and 8 logic channels with sophisticated cross-triggers.**

Spectrum Analyzer
✓ **Integrated real-time spectrum analyzer for each analog channel with concurrent waveform display.**

Logic Analyzer
✓ **8 logic, External Trigger and special purpose inputs to capture digital signals down to 25nS.**

Data Recorder
✓ **Record anything DSO can capture. Supports live data replay and display export.**

Networking
✓ **Flexible network connectivity supporting multi-scope operation, remote monitoring and data acquisition.**

Data Export
✓ **Export data with DSO using portable CSV files or use libraries to build custom BitScope solutions.**

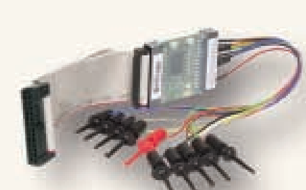
BitScope DSO Software for Windows and Linux

BitScope DSO is fast and intuitive multi-channel test and measurement software for your PC or notebook. Whether it's a digital scope, spectrum analyzer, mixed signal scope, logic analyzer, waveform generator or data recorder, BitScope DSO supports them all.

Capture deep buffer one-shots or display waveforms live just like an analog scope. Comprehensive test instrument integration means you can view the same data in different ways simultaneously at the click of a button.

DSO may even be used stand-alone to share data with colleagues, students or customers. Waveforms may be exported as portable image files or live captures replayed on other PCs as if a BitScope was locally connected.

BitScope DSO supports all current BitScope models, auto-configures when it connects and can manage multiple BitScopes concurrently. No manual setup is normally required. Data export is available for use with third party software tools and BitScope's networked data acquisition capabilities are fully supported.



www.bitscope.com

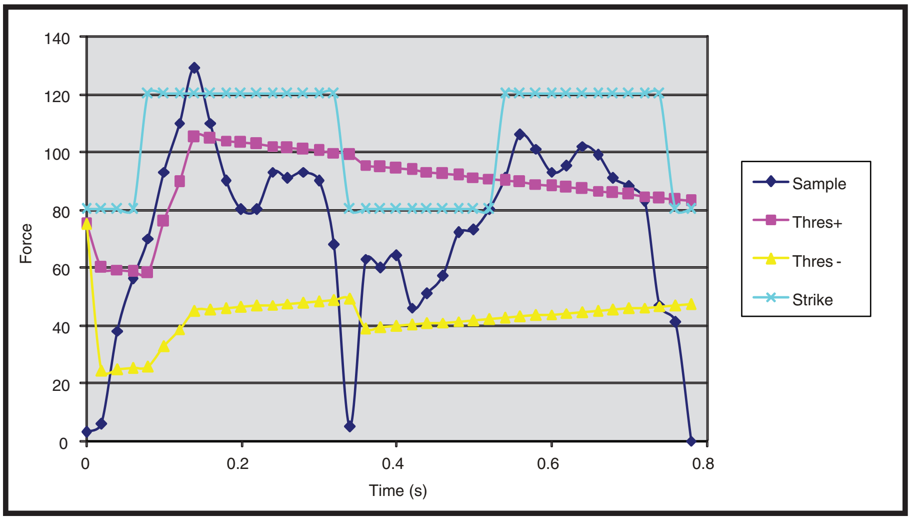


Figure 4—The sample curve shows the force data from the ADXL330 when worn on a runner's hip.

under the big electrolytic capacitors to reflow and even then I had to go back and touch up two of them with an Aoyue Tongyi Electronic Equipment 850A++ hot air rework tool. At 10 minutes, I shut the skillet off and waited for the PCB to cool (see Photo 2c).

While processing the PCB in the photo sequence, temperature data was gathered from five locations: two locations from the surface of the skillet, from the solder pad of a 1206 resistor, from the solder pad of a brown capacitor, and from the solder pad of the audio jack. The data, plotted in Figure 3, is compared against the recommended profile taken from the Kester R276 technical documents.

There are two shortcomings associated with the reflow skillet: its profile is nowhere near as steep as what is recommended by Kester and the maximum temperature is not as high as the recommended 550°F. The slope of the profile could be increased by preheating the skillet to around 200°F, putting the PCB on the skillet, and then cranking up the skillet temperature to 400°F. The lack of a peak temperature might be addressed by some "adjustments" to the skillet's thermostat. Despite these shortcomings, the solder paste and SMT components have proved to be up to the task. In practice, the reflow skillet has produced consistently excellent results for all of my students. My hats off to the folks at SparkFun Electronics for thinking outside of the box and coming up with this simple and low-cost technique.

THE PEDO-SPEEDO

To demonstrate the capabilities of the DIY Wii, let's examine one of the final projects from my embedded systems class, the pedo-speedo. The goal of the pedo-speedo was to use the

time-varying acceleration data from a DIY Wii mounted on a runner's hip to infer his speed and distance. The starting point for this project was to configure the DIY Wii as a data logger to collect data over a fixed-length course to determine the relationship between a runner's cadence and his speed. Getting the Microchip Technology MCC18 compiler to free up several blocks of its 1.5 KB of RAM proved to be unexpectedly difficult. The solution involved editing the linker file and telling the compiler to alias several contiguous blocks of RAM with a variable name, which is referenced from within your C source file (see Listing 1).

With 640 bytes of RAM, there were about 13 s of logging capability. The logged data was then downloaded and analyzed in Excel. An example of the ADC samples from the ADXL (along with a lot of other information) is plotted in Figure 4. The "sample" curve in Figure 4 represents the force at the runner's hip over two footfalls. Each footfall event is punctuated by two peaks, one generated by the heel strike and the other by the toe launch.[2] The change in amplitude between the two footfall events reflects the fact that the DIY Wii was mounted asymmetrically over one hip. The period of the footfalls was extracted from the sample data by converting it into a binary signal, the "strike" curve in Figure 4, which represented if the force was high or low. To extract the signal from the sampled data, you would normally write an algorithm, which looked for a change in slope of the force data. Clearly, this will not work with the data in Figure 4. The data contains local minima and maxima. The solution to this problem was to employ hysteresis. When the sample is greater than an upper threshold, strike goes high. When the sample is less than a lower threshold, strike goes low. Otherwise, strike retains its value. From the data in Figure 4, the upper threshold should be set to 100 and the lower threshold should be set to 40. However, when the pace of the runner changes, so do acceptable values for thresholds. The solution was to constantly look for the minima and maxima of the samples and set the upper and lower thresholds 25% over the minima and 25% below the maxima. The minima and maxima were maintained using a leaky capacitor model. In this model, the current maxima are compared to the

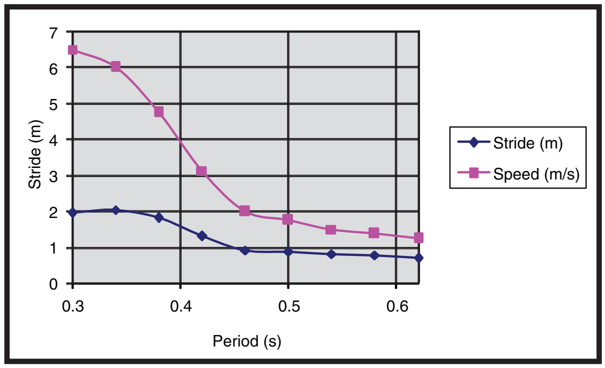


Figure 5—Here is the relationship between a runner's cadence and his stride length.

Listing 1—To get the MCC18 compiler to recognize an array larger than 256 bytes, you have to modify the linker file (top) and use `#pragma` directives in your source file (bottom).

```
//--------------- Modified 18f4520.lkr file ---------------//
ACCESSBANK NAME=accessram      START=0x0       END=0x7F
DATABANK      NAME=gpr0        START=0x80      END=0xFF
DATABANK      NAME=gpr1        START=0x100     END=0x13F
DATABANK      NAME=big         START=0x140     END=0x47F     PROTECTED
DATABANK      NAME=gpr4        START=0x480     END=0x4FF
DATABANK      NAME=gpr5        START=0x500     END=0x5FF
ACCESSBANK    NAME=accesssfr   START=0xF80     END=0xFFF     PROTECTED
SECTION       NAME=xaxis_scn   RAM=big
STACK         SIZE=0x100       RAM=gpr5

//--------------- Directive in C file ---------------//
#pragma udata xaxis_scn
static char xaxis_buffer[0x340];
#pragma udata
```

current sample. If the sample is larger than the maxima, the maxima is assigned the value of the sample. Otherwise, the maxima is decreased by a fixed amount. This way, the maxima "discharges" down from its peak values and is able to quickly conform to any decrease in the amplitude of the runner. The thres+/thresh– curves in Figure 4 are set to 75%/25% of the difference between maxima and minima. The strike signal goes high when sample is greater than thres+. The strike signal goes low when sample is less than thres–. Otherwise, the strike remains unchanged.

All of the analysis in Figure 4 was performed to determine how the period of the footfalls related to the speed and stride length of a runner. A bunch of trials and a little massaging of the data produced Figure 5.

The nonlinear relationship between the variables in Figure 5 means that it makes sense to have the pedometer application use a look-up table. This is accomplished by comparing the incoming period against the periods of the data points in Figure 5. The interpolated value is formed from the closest pair of periods. To further increase the efficiency of the computations, the decimal numbers were stored using a fixed-point representation.

The finishing touches on the pedo-speedo were to have the LCD show the instantaneous speed, distance covered, run time, and number of footfalls. The actual run time was maintained by enabling the run-time clock when the difference between thresh+ and thresh– exceeded some threshold. This way, the pedo-speedo does not measure the time you are standing around getting the mail out of the mailbox. A nice final touch was enabling the speaker to buzz when the strike signal was low. This gives the user some auditory feedback in time with his cadence and assures the user that the unit is functioning properly.

I've used the pedo-speedo on a couple of training runs and I am surprised at its accuracy. I often make spontaneous decisions about which way to go during my runs, making it difficult to determine the distance of a run. However, this is not a problem when I have the pedo-speedo with me.

MASTER SMDs

I hope that this article has made you confident enough to use surface-mount devices in your next project. With a few tools and a willingness to try, I am sure that you can pick up the necessary skills to master SMDs. Learning these skills will turn out to be a wise investment because sooner or later you will run across a device like the ADXL330, which comes only in a surface-mount package. After your first project using SMDs, I am sure that you will find that their space savings enables you to pack a big punch in your tiny embedded project.

In the next installment, I will show you how to take the acceleration values from the DIY Wii, send them to your PC, and use this information to manipulate 3-D graphics. Hopefully, you will learn how to leverage the power of your PC and the diminutive size of microcontrollers to build your own video games. ■

Chris Coulston has a PhD in Computer Science and Engineering from Penn State University. He is an associate professor and program chair of Electrical, Computer, and Software Engineering at Penn State, Erie. Chris also runs ATAN consulting, an electronics development and manufacturing firm that provides instrumentation solutions for local industry. In his spare time, he likes to ride his bicycle around Erie county and race on the weekends. You can e-mail Chris at coulston@psu.edu.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2008/212.

REFERENCES

[1] R. Lacoste, "Easy Reflow: Build an SMT Reflow Oven Controller," *Circuit Cellar* 168, 2004.

[2] J. Scarlett, "Enhancing the Performance of Pedometers Using a Single Accelerometer," Analog Devices, Inc., www.analog.com/library/analogDialogue/archives/41-03/pedometer.html.

RESOURCE

SparkFun Electronics, www.sparkfun.com.

SOURCES

ADXL330
Analog Devices, Inc.
www.analog.com

R276 No-clean solder paste
Kester, Inc.
www.kester.com

LCM-S01602DTR/M LCD
Lumex, Inc.
www.lumex.com

MCC18 Compiler, MCP1252 DC-to-DC converter, PICkit 2 development programmer/debugger, and PIC18F4520 microcontroller
Microchip Technology, Inc.
www.microchip.com

by Monte Dalrymple

Robotics Made Easy

A Peripheral Chip for Low-Level Functions

Robotics and motion control applications typically require specialized hardware and software. Monte introduces a simple peripheral chip that can take care of all of the low-level functions.

In these days of chips with almost unlimited transistor budgets, it is hard to find a new peripheral chip to use when the main CPU doesn't have exactly the right mix of I/O. This is because it's hard for chip architects to come up with a balance of functionality (transistors) versus pin count that makes sense.

As a result, system designers often end up having to add either an FPGA or a small microcontroller to perform specialized I/O tasks. While this isn't a big deal as far as the hardware is concerned, it adds another layer of software complexity.

The Rabbit Semiconductor I/O (RIO) device was designed specifically with these facts in mind. In this article, I'll describe the features of this new peripheral chip that should take care of everything you might need for a robotics or motion-control application. Then I'll give some examples of how to use the device for everything from driving motors to sensing position.

TALKING TO THE CPU

This device consists of eight identical channels in a small (10 mm × 10 mm) 64-pin package. To make the RIO as general-purpose as possible, there are six different options for connecting the device to a CPU. The signals for the different bus interface options are shown in Table 1.

A traditional parallel interface is available with the usual Chip Select, Read Strobe, Write Strobe, Address Bus, and 8-bit Data Bus. The interface is completely asynchronous, but because all of the internal logic is synchronous, there is a cycle-time restriction on the bus. A Wait output is automatically generated by the device to enforce the cycle-time restriction so no extra external logic is required.

Each of the eight channels is individually addressed, but the registers within a channel are accessed indirectly, using a two-step process that first writes the register address, followed by a read or write of the data. Global registers are accessed directly.

Using the parallel bus interface is not without its drawbacks, because six pins that are normally available for channel I/O are used for part of the data bus. This is why using the device with a serial bus interface makes the most sense.

Five different options are available for the serial bus interface. Either separate serial input data and serial output data can be selected or a bidirectional serial I/O bus can be used. With each of these options, the data can be sent either MSB first or LSB first. The clock polarity is not programmable though; output data always changes on the falling edge of the clock and input data is always sampled on the rising edge of the clock.

As in the case of the parallel bus,

Parallel interface	Parallel function	Serial interface	Serial function
SER/*PAR	Low (select parallel)	SER/*PAR	High (select serial)
*IOCS	Chip select	*IOCS	Chip select
*IORD	Read strobe	GPIN[4]	General-purpose In
*IOWR	Write strobe	GPIN[3]	General-purpose In
A[2]	Channel select 2	GPIN[2]	General-purpose In
A[1]	Channel select 1	GPIN[1]	General-purpose In
A[0]	Channel select 0	MSB/*LSB	MSB First/LSB First
G/*C	Global/Channel select	S/*RN	Serial/RabbitNet
*P/I	Pointer/Indirect	BIDI/*NORM	1-Wire/2-wire Data
*WAIT	Wait request	SERO	Serial data out
D[7]	Data Bus 7	SERCLK	Serial clock
D[6]	Data Bus 6	SERIO	Serial data in/out
D[5]	Data Bus 5	Ch6Port[3]	Channel 6 I/O
D[4]	Data Bus 4	Ch6Port[2]	Channel 6 I/O
D[3]	Data Bus 3	Ch6Port[1]	Channel 6 I/O
D[2]	Data Bus 2	Ch7Port[3]	Channel 7 I/O
D[1]	Data Bus 1	Ch7Port[2]	Channel 7 I/O
D[0]	Data Bus 0	Ch7Port[1]	Channel 7 I/O
*INT	Interrupt request	*INT	Interrupt request

Table 1—The RIO can be used with either a parallel bus interface or a serial bus interface. The parallel bus requires six of the 32 available channel I/Os.

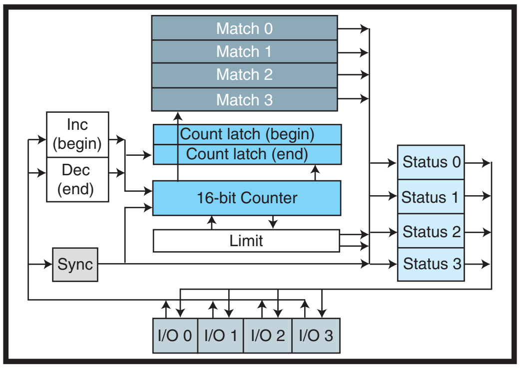


Figure 1—All eight identical channels are built around a 16-bit up/down counter and four I/O pins.

first the address and transfer direction information is written to the device and then the data is transferred in a second serial operation. The serial bus is still reasonably high-performance, because the maximum serial clock rate is the system clock divided by four and the system clock frequency can be as high as 40 MHz.

The final bus interface option uses the serial RabbitNet protocol, which is a link layer protocol for connecting a processor to distributed slave devices that is proprietary to Rabbit Semiconductor. I won't go over the details here, but you can think of RabbitNet as an inexpensive alternative to USB.

The serial interface options do not use nearly as many pins as the parallel bus option, so there are a number of pins left over. Four of the unused pins serve as general-purpose inputs in this case.

The dedicated Interrupt Request pin is independent of the bus interface. To make the device as general-purpose as possible, there is no provision for an interrupt acknowledge signal. Instead, interrupts are removed by clearing the status bits that signal the type of interrupt that is pending. Determining what kind of interrupt is pending is straightforward. A global register has an interrupt pending status bit for each channel and a register within each channel indicates exactly what type of interrupt is pending.

CHANNEL HARDWARE

Each of the eight channels in the device is built around a 16-bit up/down counter and four I/O pins. While this might not sound like much hardware per channel, about 25 8-bit control registers are required to control the operation of each channel. Figure 1 shows a basic block diagram of a channel.

Although the hardware of the counter is 16 bits wide, the actual count limit can be set to any value and the counter wraps around while counting in either direction. This is very useful when you need something that counts by other than a power of two. I'll talk about one example when I get to the Position Sensing section.

Four sets of compare logic and four 16-bit match registers are connected to the counter. Each comparator signals a match when the count is identical to the value in the match register. Each match signal can generate an interrupt and be used to set or reset any of the four channel status bits.

Each of the four channel status bits can be output on an I/O pin. The status bits can be set or reset by a counter rollover, in either direction, or by a sync signal (more on that later). This enables the creation of PWM or pulse position modulation (PPM) signals. A status bit can also be ANDed with one of the least significant bits of the counter to create a pulse train that can be output on an I/O pin.

While the status bits are the primary outputs for the I/O pins, the I/O pins can also be used as simple outputs, sequenced outputs, simple inputs, or counter inputs. Each I/O pin is individually programmed. The sequenced output option enables rudimentary waveform creation by stepping through one of four programmed values on each counter rollover.

The counter obviously needs increment and decrement signals. There are a number of options available for each. The increment and decrement signals can come from any channel I/O pin and can be either level- or edge-triggered. There is a special quadrature decoding option for increment/decrement and these signals can also be forced under program control.

The counter can be used for timer applications in a variety of modes. In this case, the increment and decrement signals function as begin and end signals, either starting and stopping the counter or latching the current count when it is free-running. Of course the current count and both of the latched counts can be read by the CPU.

In addition to the increment and decrement signals, each channel also has a sync signal, which is used to reset the counter to all zeros. Like the increment and decrement signals, the sync signal has a range of options available and is very useful for synchronizing counters in different channels or synchronizing the counter to an external event. Note that the sync signal can be forced under CPU control.

That's the basic functionality of each channel. Now let's look at how to put the RIO to work in motion control and robotics applications.

MOTORS AND H-BRIDGES

The most common way to power a DC motor that needs to run in either direction is with an H-bridge. An example of an H-bridge using MOSFET transistors is shown in Figure 2. Other types of drivers are also possible, ranging from bipolar transistors to relays. But it's the circuit topology that's important for this discussion, not the exact characteristics of the drivers used. The four signals needed for an H-bridge are the main reason that there are four I/O pins per channel in the device.

To drive the motor in one direction, circuit legs A and D are enabled, providing a power-to-ground path in one direction through the motor. For motor rotation in the opposite direction, circuit legs B and C are enabled, providing current flow in the opposite direction through the motor.

The motor will run at full speed while the current is continuous. PWM is normally used to vary the speed, if required. This works because the

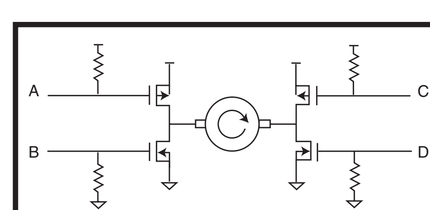


Figure 2—Driving a motor in both directions is usually done with an H-bridge. But don't ever turn on both drivers on the same side.

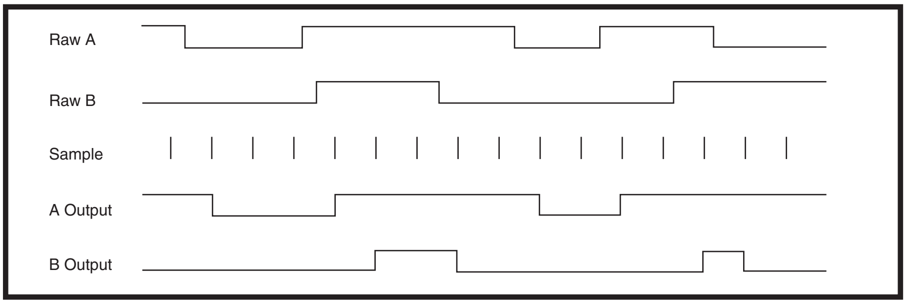


Figure 3—The pin-pair protection guarantees that both drivers on the same side of an H-bridge will never be on at the same time. It also guarantees a dead time between changing the motor direction.

motor inherently acts like a low-pass filter for the pulse-width-modulated current.

There are two issues with an H-bridge that complicate its use. The first should be obvious from the circuit topology. If circuit legs A and C (or B and D) are inadvertently active at the same time, no current goes through the motor. Instead there is a direct power-to-ground short through the drivers, which will almost certainly overheat and then destroy one or both of the drivers.

The second issue becomes a problem as a result of the first. Motors usually require a fair amount of current and this means that the drivers don't turn on and off very quickly. So it isn't enough to guarantee that both drivers on one side of the H-bridge are not on at the same time; they must also have some non-overlap time.

The RIO contains special circuitry that can be used to protect the user from inadvertently running afoul of either of these issues. This function is called pin-pair protection and is controlled via a pair of global registers outside the normal channel register set.

After a device reset, all of the channel I/O pins are inputs. If the four I/O pins of a particular channel are connected to the H-bridge in Figure 2, the state of the four I/O pins will be the default all-off state because of the resistors. Enabling the pin-pair protection for the channel pins connected to the left and right legs of the H-bridge samples this state and saves it for use by the pin-pair protection circuitry.

From that point on, until the next device reset, the complement of the stored state (which would turn on both drivers) is guaranteed never to be driven on that pair of pins. The circuitry that does this is actually the last bit of logic before the pad cells for the pair of I/O pins. If the channel somehow tries to output this illegal value, the pin-pair protection circuit substitutes the stored all-off value instead.

There is more to the story because the protection circuitry runs off of an independent timer that guarantees a dead time for the H-bridge drive signals. The timer is used solely for the pin-pair protection. An example of how it works is shown in Figure 3.

The timer samples the channel outputs and transfers legal values to the I/O pins while substituting the all-off value for an illegal value. All of the features, from the four I/O pins with independent PWM signals to the pin-pair protection circuitry, make driving a motor with an H-bridge painless and safe.

POSITION SENSING

Trying to control any kind of motion without some type of feedback about position or velocity is a recipe for disaster. One of the most common ways of sensing the position of anything that rotates is with a pair of signals generated by a quadrature encoder. Figure 4 shows the operation of quadrature signals.

The phase relationship between two quadrature signals provides direction information, while the frequency of the signals provides rotational velocity information. When a separate index pulse is available, counting the edges on the quadrature signals enables absolute position to be determined.

Quadrature encoders are available as off-the-shelf components with a variety of resolutions. The most common encoders have either 500 or 512 quadrature cycles per revolution. This is where the programmable count limit of the channel counter helps, because it allows any quadrature cycle count to be supported. The count limit should be set to four times the cycle count, minus one, because the counter increments or decrements on each valid edge of one of the quadrature signals.

As with the H-bridge, there are some subtle issues involved in decoding quadrature signals. The first is the handling of invalid transitions. As shown in Figure 4, only one quadrature signal should change at a time. Two signals switching at the same time is illegal.

The second issue has to do with how the signals are created. Because they are most often the outputs of photodiodes or other devices that respond to light and that are being interrupted by a mechanical device, the signals often have less than ideal switching characteristics. In fact, if the rotation of the encoder is stopped right where a transition is imminent, one output may oscillate.

The RIO automatically takes care of these two issues. The logic ignores all illegal transitions by holding the state of the counter unless a valid transition has been detected. In addition, the quadrature inputs both contain digital

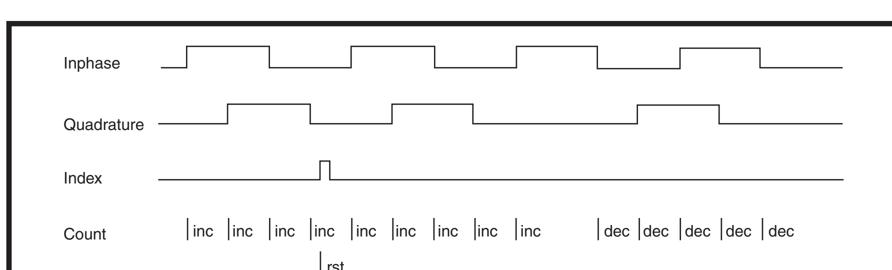


Figure 4—Rotational position and direction is sensed using quadrature signals. Tracking the absolute position requires an index pulse.

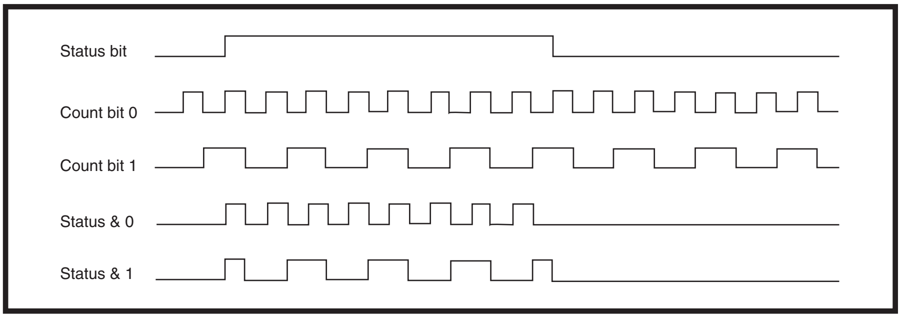


Figure 5—Driving an ultrasonic transducer directly requires a burst of pulses. This type of output can also be used to reduce the filtering requirements for a PWM signal used to create an analog value.

filters that feed clean signals to the actual decoder.

Measuring the rotation of a shaft is one thing, but what about sensing position relative to the rest of the world? Contact sensors are easy, because the state of the I/O pins can be read at any time. If you prefer an interrupt, just assign an I/O pin as the increment or decrement signal and select the appropriate edge or level to generate an interrupt.

Non-contact position sensing is supported by the Toggle Output mode. In this mode, an I/O pin can be driven with the logical AND of a status bit and one or more of the least significant bits of the counter. This feature is useful for creating a tone burst to drive an ultrasonic transducer (see Figure 5).

One channel can handle both the creation of the tone burst and measurement of the time to detect an echo. The sequence can be started under program control by forcing a sync signal, which can set the status bit and enable both counting and the burst of pulses. A second I/O pin can output the range gate signal, created from another status bit. A third I/O pin is used to input the echo detect signal. The input is programmed as the decrement signal to latch the count for determining the round trip delay.

THE LITTLE STUFF

There are a number of other bits and pieces that are common in robotics and motion control. The first is servos to move things. The most common servos use a PWM signal to control movement. Typical timing for a servo is shown in Figure 6.

The 20-ms frame rate of a servo coupled with the 16-bit counter in a channel implies a maximum frequency into the counter of just over 3 MHz. Because the RIO is specified to operate at up to 40 MHz and the CPU bus interface operates off of the clock input, this could be a problem.

To handle this situation, there is a global 8-bit prescaler for the clock that is fed to the channels. The prescaler decouples the bus speed from the channel clock rate and is individually enabled for use by each channel so not all channels have to be limited by the slow clock.

Saelig
UNIQUE PRODUCTS + SUPPORT!
Scopes & Wfm Gens
20/40MHz AWG
NEW! RIGOL
25/40MHz USB 14-bit ARB w. USB RS-232, LAN/GPIB. 10 built-in wfms; FM, AM, PM, FSK, SWEEP and BURST; 6 digit 200MHz frequency counter.
DG2021A/DG2041A $995/$1495
100MHz DSO
NEW! RIGOL
100MHz 2-ch 400MS/s DSO 2000 wfm/s refresh rate. Ultra Compact Design with small footprint saves bench space.
DS1102C $999
100/200/300MHz DSO
NEW! RIGOL
100/200/300MHz 2 ch, 2GS/s DSO 2000 wfm/s refresh rate. Ultra Compact Design with small footprint saves bench space. DS1102A $1295
DS1202A $1595
DS1302A $1995
Scope + Analyzer
NEW! RIGOL
100MHz 2-ch /16 logic scope and logic analyzer. 2000 wfm/s refresh rate. See waveforms and logic data simultaneously. Trigger on data or signal edge.
DS1102CD $1499
15/20MHz Wfm Gen
NEW! RIGOL
15/20MHz USB 14-bit Function/ Arbitrary wfm Generators. 10 built-in wfms; FM, AM, PM, FSK, SWEEP and BURST; 6 digit 200MHz frequency counter.
DG1011 / DG1021 $650 / $795
Announcing
...a new name in high-end, high reliability, value-priced test gear: RIGOL. (pronounced 'regal') Formerly the partner of a large scope company we can't name, Rigol produces the best quality/value LCD scopes available on the market today.
Color LCD Scope
Best Seller
Owon - 25MHz 2-ch + trigger standalone USB bench scope with 7.8" color LCD. Battery powered optional.
PDS5022S $325
PDS6062T $599
Low-Cost Scope
2-ch 40/100/200MS/s 8-bit scope range with 5/10/25MHz bandwidth and USB2.0 for fast screen updates. Inc. scope/FFT /logging software, case, probes.
PS2203/4/5 from $318
PicoScope 3206
PS3206 - Is an impressive 2-ch 200MS/s 8-bit PC scope adapter - 10GS/s for repetitive signals! No power supply needed, simple USB 2.0 connection
PS3206 Bundle $1574
2/4ch 12-bit USB Scope
Pico - New 2/4-ch 12-bit 20MS/s USB2.0-connected/powered scope adapters. Scope, FFT, and meter displayed at once with included software.
PS3424 / PS3224 $1398 / $793
World's Fastest
Winner!
PS5204 - 2-ch 250MHz Scope with 128 MS/s and real-time sampling of 1GS/s. Adv triggering. 250MHz spec. analyzer and 125MS/s waveform gen.
PS5203 32MS buffer $2237
PS5204 128MS buffer $3360
60/100/120MHz AWG
NEW! RIGOL
60/100/120MHz USB 14-bit ARB w. USB RS-232, LAN/GPIB. 10 built-in wfms; 6 digit 200MHz frequency counter; opt. 16-ch gen. DG3061A $1895
DG3101A/DG3121A $2795/$3995
6 in 1 Scope
6-in-1 USB scope adapter! 200kHz 2-ch 10-bit scope, 2-ch spectrum analyzer, 2-ch chart recorder, 16-ch 8MHz logic analyzer, 5-ch signal gen, 8-ch pattern gen. SPI/I2C/1-w/UART decoding.
PoScope with probes $199
Automotive Testing
Automotive Diagnostics - Award-winning Pico kits turn your PC into a powerful vehicle-electrics diagnostic tool. 2 & 4 ch. versions for components & circuits.
PS3423 automotive kit $2293
Mixed -Signal PC Oscilloscope
100 MHz Scope and Logic Analyzer lets you do complex triggering to find hard-to-get-at glitches, spikes, etc. Huge 4 or 8 MS buffer for deep data drilling and zooming. Optional built-in swept signal generator. 2 Analog channels x 10, 12, 14 bit with more than 60 dB dynamic range. 8 digital inputs for mixed signal display/trigger.
CS328A-4 (4MS Buffer) $1149
CS328A-8 (8MS Buffer) $1642
CS700A (signal generator) $249
Handheld Scope
20MHz or 60MHz rugged handheld USB scope with 3.8" color LCD. Built-in meter - great for your tool kit.
HDS1022M (20MHz) $499
HDS2062M (60MHz) $699
Pen Scope
PS2105 - 25MHz USB powered scope-in-a-probe! Up to 100MS/s, 24kS buffer C/VB/ Delphi/LabView/VEE drivers.
PS2105 (25MHz, 100MS/s) $372
PS2104 (10MHz, 50MS/s) $234
I2C
USB to I2C
Devasys - Provides a simple "drop-in" solution for connecting your PC to 90kHz I2C + 20 I/O lines. Free software. Use multiple boards for more I2C/IO.
USB I2C/IO $79
Ethernet to I2C
NM7010A-LF REV 2 - Complete, hardwired TCP/IP-I2C module. Offloads stack for hi-speed Internet-enabling microprocessor systems. Drop-in W3100A-based board for instant network speeds to 8Mbps. $22
Ethernet to I2C
Fiveco - TCP/IP server with I2C, 19 x digital I/O, 5 x 10-bit A/D inputs, and UART serial interface. Connect I2C devices/sensors to an Ethernet network!
FMod-TCP from $250
I2C for PCs
PCI93LV: industry-standard I2C card for PCs. WINI2C/PCI software gives windows-interface to develop and debug I2C bus systems. UCA93LV is USB version. Transparently monitor at 400kHz! $499
Blue & White LCD
Comfile - LED backlit 4 x 20 character display. CuNET, RS232 or I2C communication to 115Kbps. BASIC programmed. 2.36" x 3.86"
CLCD420-B $49
Saelig
Alan Lowne says: "I started Saelig as an engineer - for engineers! Saelig - an Olde English word meaning 'happy, prosperous, blessed' - which is what I want for my customers, my staff, and myself! See my blog for behind-the-scenes company details, business hints, kudos, etc. "
Ethernet
Serial-Ethernet Cable
Network your serial product easily without a PC using this 28" cable. Provides an instant IP address. RS-232 port compatible with most 10/100mpbs Ethernet hubs.
eCOV-110-P $89
Ethernet to Serial
WIZnet - Ready-to-go serial-to-Ethernet Gateway. Simple set-up for 10/100Mbps Ethernet - 230kbps serial. Ethernet/ Internet-enable any serial part.
EG-SR-7150MJ $34
Ethernet-IO
Ether-IO - UDP/IP-controlled 24 digital I/O board 3 x 8-bit TTL ports each independently programmable. Connects to any TCP/IP Ethernet network.
Ether-IO 24 $119
Ethernet-IO
Fiveco - TCP/IP server with I2C, 19 x digital I/O, 5 x 10-bit A/D inputs, and UART serial interface. Connect I2C devices/sensors to an Ethernet network!
FMod-TCP from $250
Serial Server
RS232/422/485 device server for network-enabling legacy serial devices. 1/2/4/8/16/32 ports 10/100Mbps-230kbps. Static IP/DHCP/PPPoE.
Portbase-3010/RS232 from $149
Mini Web Server
PicoServTM Miniature Web Server allows remote I/O and servicing. Collect analog & digital data over a web connection. Remote sensing/ control, SCADA, building control.
PP1001-V00 $205
Misc
EMC Spectrum Analyzer
Handheld PalmPC-based 2.7GHz Spectrum Analyzer. Continuous/ single/peak-hold/avg sweeps - unlimited storage for wfms, set-ups, etc. Built-in Wi-Fi/Bluetooth/IR.
PSA2701T $1990
DMM
NEW! RIGOL
6 1/2 digit 50KS/s, 2MS mem. Opt. 16-ch mux. Automeasure DC/AC, 2/4-wire R, C, Continuity, Diode, Frequency, Period, Ratio, Temp, Sensor, High/Low Limit Math.
DM3061 $795
USB Bus Analyzers
NEW!
USB1.1/2.0/WiMedia analyzers and now - EX260 can test USB3.0 5Gbps too! Emulate host/device, verify compliance.
TR110 / EX200 / EX260
$999 / $2999 / CALL
Wireless Modules
Easily create mesh networks at 910MHz - like Zigbee only better! Complete RF solution for fast & reliable data transmission.
B915FHtiny-plus/40-SMD-WA $64
B915FHtiny-plus/40-DIP-EA $804
1/2/4/8/16 x RS232
Add COMports via your PC's USB Port. USB-COM-S $20
USB-2COM $48
USB-4COM $105
USB-8COM $195
USB-16COM-RM (rackmount) $459
SM PCB Adapters
OM-Adapt SM miniboards have two footprints on each side. Now use your ultra fine pitch SMD components with more useful 0.1" inline spaced holes.
One-to-one pinouts. from $9.52
Saelig.com
unique electronics
888-7SAELIG info@saelig.com
We search the world for unique, time-saving products - see our website for 100s more: volt/temp data loggers, wireless boards, LCD display kits, Ethernet/IO, USB/RS232/485, USB-OTG, instant Ethernet-serial, CAN/LINbus, USB cables/extenders, line testers, logic analyzers, color sensors, motion controllers, eng. software, wireless boards, SMD adapters, I2C adapters, FPGA kits, GPS loggers, automotive testing, security dongles, video motion detectors, crystals/oscillators, custom switches, barcode scanners, robots, DSP filters, PLCs, Remote MP3 players. etc.
Don't forget to ask for your FREE Starbucks card with order! Check www.saelig.com often for special offers and bargains.

The channel I/O pins can be used as simple inputs and outputs, but the counter makes a number of other functions possible. For example, an input can start the counter operating as a timer, with match values used to create a time-delayed output pulse or two. A delayed interrupt can also be created with this configuration.

The counter operating as a timer can be used to create synchronized output signals or special phase relationships among output pins. The global sync signal enables multiple channels to be synchronized when more than four pins need coordinated timing.

ALL TOGETHER NOW

Figure 7 shows an example of how the RIO might be configured to control the operation of a basic robotics platform. For this example, I've assumed a track drive with a rotating turret to hold sensors and actuators. I've also assumed a serial bus interface so the entire channel I/O is available.

Channels 0 and 1 are used to control H-bridges for the left and right tracks.

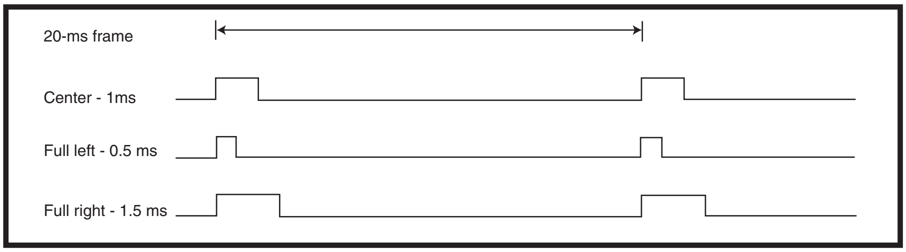


Figure 6—Many servos use a 20-ms frame and a 1- to 2-ms position pulse. It's easy to create four such signals per channel.

No positional feedback is required for the tracks, but velocity information is required for closed-loop control and fault detection. Two inputs and the timer in channel 2 are used for this function. The timer can be programmed to measure either the width of the velocity pulses or the time between successive velocity pulses, depending on how the signals are created. The two remaining pins of channel 2 are available for use as simple inputs or outputs.

Channel 3 is used to control an H-bridge that drives the motor for the turret. Controlling the turret requires positional feedback, so channel 4 is used as a quadrature decoder for this function. The remaining I/O pin in channel 4 feeds back the low battery detect signal or some other bit of status information.

Channel 5 is used to control four servos, perhaps for four degrees of freedom on an arm attached to the turret. This channel uses a divided clock for the slow frame rate of the servos.

Channel 6 controls a pair of servos for tilt and pan for a camera on the turret, while the other two pins connect to two contact sensors for the inevitable case when the platform bumps into something. The two inputs can be either polled or used to generate interrupts. The divided clock is also used for this channel.

Channel 7 is used for the last two contact sensor inputs, so two pins and the timer are available for some other use. If the contact sensor inputs are polled, they could be assigned to the unused pins on channel 2, freeing up this channel for some other function.

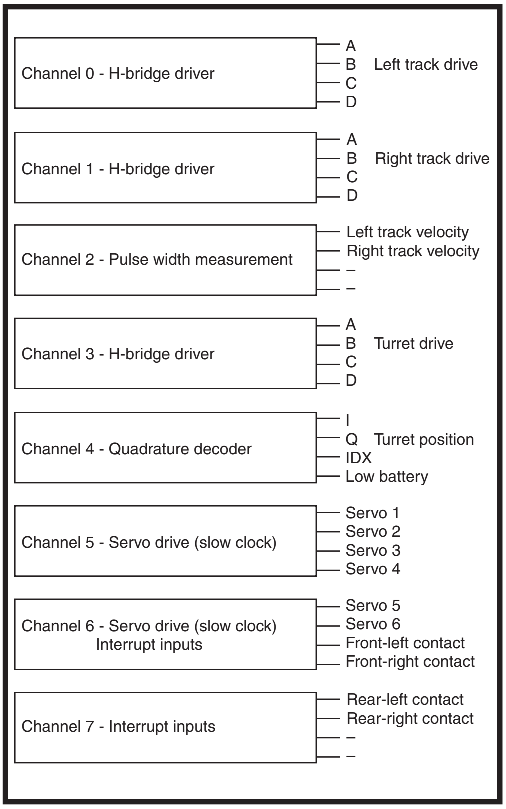


Figure 7—The RIO has the resources to handle an entire robotics application, with room to spare.

WRAPPING UP

At first glance, the RIO seems like a pretty simple peripheral chip, but a fair amount of thought went into the features to make the device useful for robotics and motion control applications. Yes, everything that the RIO does can also be done with an FPGA or a small microcontroller, but dedicated hardware will almost always be a better choice. Give it a look next time you need to control something that moves.

Author's Note: Thanks to Norm Rogers of Rabbit Semiconductor for his vision of a simple multi-purpose I/O chip. I also want to thank Pedram Abolgasem, Lynn Wood, and Steve Hardy for helping to bring it to fruition.

Monte Dalrymple has been designing integrated circuits for 29 years. After tiring of the corporate world, he has been designing on a contract basis for the past 12 years. Monte is the designer of all four generations of Rabbit microprocessors as well as the Rabbit I/O chip. You may reach him at monted @systemyde.com.

RESOURCES

Rabbit Semiconductor, "RIO Datasheet," www.rabbitsemiconductor.com/products /rio/Rabbit_RIO.pdf.

———, "RIO User's Manual," 019-0158 070718-C, www.rabbitsemiconductor. com/documentation/docs/manuals/ RabbitRIO/RabbitRIO.pdf.

SOURCE

RIO Programmable I/O chip
Rabbit Semiconductor
www.rabbitsemiconductor.com

Efficient Micro Mathematics

Multiplication and Division Techniques for MCUs

Most inexpensive microcontrollers don't have a hardware multiplier module and typically require numerous instruction cycles to perform multiplication and division operations. Kripasagar describes techniques based on Horner's method for performing efficient multiplication and division in the absence of a hardware multiplier.

Low-cost microcontrollers are typically targeted at applications with low levels of complexity and are optimized for cost and performance. The devices should offer low power consumption and be simple to design. They should also provide easy interfaces to external peripherals. As technology advances, the demand for processing capabilities and efficiency increases. This demand calls for an upgrade of existing microcontrollers with introduction of newer peripherals. The design tools to support enhanced hardware and software must also go through a process of considerable development. Manufacturers continuously strive to provide on-chip solutions for complex algorithms, thus increasing cost. Some microcontrollers choose to go through limited enhancements and adhere to low cost and low power consumption. Complex algorithms and functions, such as digital filtering, at times become impossible on these devices. This forces designers to search for efficient methods and make do with what is available. This article focuses on one such efficient method for microcontrollers.

ALGORITHMS

Processors are broadly classified as fixed point or floating point. Fixed-point processors support only integers, whereas floating-point processors have additional circuitry to support integers and fractions. Various standards for floating-point formats have been established to maintain uniformity across processors and their associated tools. All microcontrollers fall under fixed-point processors and are mostly 8-bit or 16-bit devices. Fixed-point processors suffer from the effects of finite word length, round-off, and truncation.[1] These issues have a direct impact on the accuracy of the results obtained during mathematical operations. A hardware multiplier is a module that supports multiplication and multiply and accumulate (MAC) operations via dedicated central processing unit (CPU) instructions. Most low-cost microcontrollers do not have a hardware multiplier module. They usually require a lot of instruction cycles to perform multiplication using alternate algorithms.

Several algorithms have been devised for fast multiply and divide using only shift and add instructions for fixed-point processors.[2] These algorithms are specific to integer-integer multiplies or divides. This does not mean that they are unable to support multipliers or divisors that are pure fractions or real numbers. The solution is a form of scaling that converts all real numbers to integers. Different standards—such as Q formats—have been introduced to accomplish this on fixed-point machines. There is, however, a potential loss in accuracy with such formats due to truncation of the real number on registers with fixed widths. Horner's algorithm attempts to reduce this error and improve accuracy. This method for multiplication is easily extended to division, which is a multiplication by the divisor's reciprocal. An innovative scaling-free method to implement integer-real multiplications will be described in this article.

Horner's algorithm is based on the

$$
\begin{array}{l}
0.2468 \times 0.1357 = \left(0.001111110010111_b\right)\left(2^{-3} + 2^{-7} + 2^{-9} + 2^{-11} + 2^{-12} + 2^{-13} + 2^{-14}\right) \\
= 0.0000011111\ 10010_b \\
+\ 0.0000000001\ 11111_b \\
+\ 0.0000000000\ 01111_b \\
+\ 0.0000000000\ 00011_b \\
+\ 0.0000000000\ 00001_b \\
+\ 0.0000000000\ 00000_b \\
+\ 0.0000000000\ 00000_b \\
\hline
\quad 0.000010001000100_b = 0.03332519\ 53125
\end{array}
$$

Figure 1—This is the binary equivalent of a 15 × 15-bit fractional multiplication with the multiplier known in advance. Dedicated code would replace each of these steps to perform a multiplication on a 16-bit CPU.

$$
\begin{aligned}
\text{Final product} &= x_6 \times 2^{-3} = \left(x_5 \times 2^{-4} + x\right) \times 2^{-3} \\
&= \left(\left(x_4 \times 2^{-2} + x\right) \times 2^{-4} + x\right) \times 2^{-3} \\
&= \left(\left(\left(x_3 \times 2^{-2} + x\right) \times 2^{-2} + x\right) \times 2^{-4} + x\right) \times 2^{-3} \\
&= \left(\left(\left(\left(x_2 \times 2^{-1} + x\right) \times 2^{-2} + x\right) \times 2^{-2} + x\right) 2^{-4} + x\right) \times 2^{-3} \\
&= \left(\left(\left(\left(\left(x_1 \times 2^{-1} + x\right) \times 2^{-1} + x\right) \times 2^{-2} + x\right) \times 2^{-2} + x\right) \times 2^{-4} + x\right) \times 2^{-3} \\
&= \left(\left(\left(\left(\left(\left(x \times 2^{-1} + x\right) \times 2^{-1} + x\right) \times 2^{-1} + x\right) \times 2^{-2} + x\right) \times 2^{-2} + x\right) \times 2^{-4} + x\right) \times 2^{-3} \\
&= x \times \left(2^{-3} + 2^{-7} + 2^{-9} + 2^{-11} + 2^{-12} + 2^{-13} + 2^{-14}\right)
\end{aligned}
$$

Figure 2—This is an example of a back substitution in the steps of Horner's method for a fractional multiplier. This is done to verify if Horner's method conforms to the conventional binary multiplication routine for fractions.

position of the bits with a value of 1 and their distance to the neighboring 1 in a multiplier. For this method to work, the multiplier or divisor should be known in advance. This method also relies on dedicated code for any multiplies or divides with a potential increase in code size. These are not serious limitations in applications where speed is of prime concern or when the multiplier or divisor does not change run-time. The canonical signed digit (CSD) format is introduced to further reduce the CPU overhead.

FILTERING

Filtering forms the crux of many digital processing algorithms. Filtering can be viewed as a weighted multiply and accumulate (MAC) process. Digital filters come in two flavors: finite impulse response (FIR) and infinite impulse response (IIR). A mathematician's view of a FIR is a transfer function with only a numerator polynomial. An IIR filter is a transfer function with both numerator and denominator polynomials. FIR filters use only present and past samples of the input signal, whereas IIR filters in addition use previous samples of the output.[1] Stability of these filters is extremely important to preserve the sanity of the signal being filtered at the output. A FIR filter is inherently stable with no restriction on the roots (also known as zeros) of its polynomial. It can also exhibit linear phase to reduce the phase distortion of any input signal. The IIR filter performs much better and requires a significantly smaller order compared to a FIR filter for the same set of filtering specifications. An IIR filter is stable only if the roots of its denominator polynomial (also known as poles) has a magnitude of less than one. Special care must be taken when scaling the IIR filter coefficients as it could easily render a stable IIR filter unstable. Digital filter coefficients almost always are real numbers. Their implementation on fixed-point processors must support scaling to integers and additional software to track any sort of overflow associated with them.[3] An alternative to all of this is to use high-level languages such as C along with a floating-point library to support integer-real number multiplies. A C floating-point library would produce accurate results but with increased CPU overhead, rendering real-time processing would be at times impossible.

Texas Instruments's ultra-low-power MSP430 microcontrollers are an example of modern 16-bit microcontrollers that support single-cycle shift and add instructions.[4] Although some of the devices offer a hardware multiplier module, this method can still be used to accomplish an integer-real multiply. Comparisons of Horner's method and Horner's method using CSD are made with existing integer-integer multiply algorithms and a C floating-point library. Results are shown to exhibit good accuracy under low-power CPU cycle count to imply reduced CPU bandwidth and lower power consumption. Performance of digital FIR digital filters on the MSP430 has also been shown to have excellent frequency responses.

FRACTIONAL MULTIPLIER

The data to be processed is usually an analog signal converted to digital samples using an on-chip ADC, for example an ADC12, a 12-bit ADC on the MSP430. The digital samples can represent analog signals such as temperature captured by a sensor or audio signals that need some sort of digital filtering.

Horner's algorithm is explained in two parts, pure fractional multipliers and pure integer multipliers. Steps have been shown to distinguish their procedures to achieve efficient multiplies.

A fractional multiplier M falls in the range $-1 < M < 1$. Multiplication by such a number can be accomplished on fixed-point machines only by scaling M an integer. Alternatively, the bits of value 1 can be identified and shift and add operations on any multiplicand x can be done. This approach would lead to a dedicated code for each multiplier. Consider the example with $x = 0.2468$ and $M = 0.1357$ with 15-bit resolution:

$$
\begin{aligned}
x &= 0.2468 = 0.001111110010111_b \\
M &= 0.1357 = 0.001000101011110_b
\end{aligned}
\qquad [1]
$$

Figure 1 shows the exact bit-wise binary addition for this multiplication.

The correct result for this multiplication using floating-point math is 0.03349076 and the absolute error

$$
x_1 = \begin{cases} \,0.000111111001011_b \\ +\,0.001111110010111_b \\ \hline \,0.010111101100010_b \end{cases}
\qquad
x_2 = \begin{cases} \,0.001011110110001_b \\ +\,0.001111110010111_b \\ \hline \,0.011011101001000_b \end{cases}
$$

$$
x_3 = \begin{cases} \,0.001101110100100_b \\ +\,0.001111110010111_b \\ \hline \,0.011101100111011_b \end{cases}
\qquad
x_4 = \begin{cases} \,0.000111011001110_b \\ +\,0.001111110010111_b \\ \hline \,0.010111001100101_b \end{cases}
$$

$$
x_5 = \begin{cases} \,0.000101110011001_b \\ +\,0.001111110010111_b \\ \hline \,0.010101100110000_b \end{cases}
\qquad
x_6 = \begin{cases} \,0.000001010110011_b \\ +\,0.001111110010111_b \\ \hline \,0.010001001001010_b \end{cases}
$$

Figure 3—This is a binary representation and mapping of the implementation steps of Horner's method for Equation 1. The result of the intermediate steps x_1 to x_6 shows its progression to the final result.

using the conventional method is 0.0001655646875, an error of approximately 5.4 LSB. This error is due to finite word-length effects on the multiplier. Horner's algorithm attempts to reduce this error.

Horner's algorithm identifies the positions of bits with a value of 1 in the multiplier and their distance to the nearest 1 to the left. This is done starting from the rightmost 1 and moving left to the last 1 before the binary point. For Equation 1 with M = 0.1357, the position of the bits with a value of 1 in the multiplier are $\{2^{-14}, 2^{-13}, 2^{-12}, 2^{-11}, 2^{-9}, 2^{-7}, 2^{-3}\}$. The distance of the closest binary 1 to the left for each of the bits is {1, 1, 1, 1, 2, 2, 2, 4}. Once this has been established, Horner's algorithm generates a set of design equations using only shift and add operations. The design equations are written in terms of the multiplicand x. It is assumed that the reader is aware that 2^{-1} is a right shift by 1 and 2^1 is a left shift by 1.

The first step is to initialize the intermediate result to x and proceed to the rightmost bit (2^{-14}). The nearest 1 to its left is at bit position 2^{-13}. The difference in weight 2^{-1} is applied to the intermediate result. The multiplicand x is then added to the weighted result for the occurrence of the 1 at bit position 2^{-13}. The result of this addition is now stored as the intermediate result x_1 for the next step:

$$x \times 2^{-1} + x = x_1 \qquad [2]$$

The next step is to proceed to the next bit with a value of 1 (2^{-13}). The nearest 1 to its left is at bit position 2^{-12}. The difference in weight 2^{-1} is applied to the intermediate result. The multiplicand x is again added to the weighted result for the occurrence of the 1 at bit position 2^{-12}. The result of this addition is now stored as the intermediate result x_2 for the next step:

$$x_1 \times 2^{-1} + x = x_2 \qquad [3]$$

The third step is to proceed to the next bit with a value of 1 (2^{-12}). The nearest 1 to its left is at bit position 2^{-11}. The difference in weight 2^{-1} is applied to the intermediate result. The multiplicand x is again added to the weighted result for the occurrence of the 1 at bit position 2^{-11}. The result of this addition is now stored as the intermediate result x_3 for the next step:

$$x_2 \times 2^{-1} + x = x_3 \qquad [4]$$

The fourth step is to proceed to the next bit with a value of 1 (2^{-11}). The nearest 1 to its left is at bit position 2^{-9}. The difference in weight 2^{-2} is applied to the intermediate result. The multiplicand x is again added to the weighted result for the occurrence of the 1 at bit position 2^{-9}. The result of this addition is stored as the intermediate result x_4 for the next step:

$$x_3 \times 2^{-2} + x = x_4 \qquad [5]$$

The fifth step is to proceed to the next bit with a value of 1 (2^{-9}). The nearest 1 to its left is at bit position 2^{-7}. The difference in weight 2^{-2} is applied to the intermediate result. The multiplicand x is again added to the weighted result for the occurrence of the 1 at bit position 2^{-7}. The result of the addition is stored as the intermediate result x_5 for the next step:

$$x_4 \times 2^{-2} + x = x_5 \qquad [6]$$

At step six, proceed to the next bit with a value of 1 (2^{-7}). The nearest 1 to its left is at bit position 2^{-3}. The difference in weight 2^{-4} is applied to the intermediate result. The multiplicand x is again added to the weighted result for the occurrence of the 1 at bit position 2^{-3}. The result of this addition is now stored as the intermediate result x_6 for the next step:

$$x_5 \times 2^{-4} + x = x_6 \qquad [7]$$

The seventh step is to proceed to the last bit with a value of 1 (2^{-3}). Because it is the last binary 1, it does not have any ones to the left; therefore, only its bit position is applied as the weight to the intermediate result to give the final product (i.e., 2^{-3}). The result is the final product:

$$\text{Final product} = x_6 \times 2^{-3} \qquad [8]$$

The procedure can be validated by back substitution to give the same result as the conventional multiply (see Figure 2).

The bit-wise realization of design Equations 2 through 7 is shown in Figure 3. It indicates the exact operations at each stage and gives a brighter picture of Horner's method. The final product is 0.000010001001001_b.

This results in an absolute error of 0.000012976796875, an error of approximately 0.42522368 LSB. Thus, Horner's algorithm is extremely accurate and does not suffer much from finite word length effects. The design equations involve only right shifts and add operations making it an extremely fast algorithm. The design equations are unique for this multiplier and a 15-bit register width. This also implies that the multiplicand x can be any number (integer/fraction) of any sign (positive/negative). If the multiplier is a negative number, 2's complement format should be used for its binary representation and should follow similar steps to obtain the design equations.

$$\begin{aligned}
\text{Final product} &= x_3 \times 2^0 = \left(x_2 \times 2^2 + x\right) \\
&= \left(\left(x_1 \times 2^1 + x\right) \times 2^2 + x\right) \\
&= \left(\left(\left(x \times 2^3 + x\right) \times 2^1 + x\right) \times 2^2 + x\right) \\
&= x \times \left(2^0 + 2^2 + 2^3 + 2^6\right) = x \times 77
\end{aligned}$$

Figure 4—Here you see back substitution in the steps of Horner's method for an integer multiplier. This is done to verify if Horner's method conforms to the conventional binary multiplication routine for integers.

$$\begin{aligned}
M &= 0.1357 = 0.00100010101\underbrace{1111}_{\text{group}}0_b \\
&= 0.00100010\,\underbrace{11}_{\text{group}}\,000\bar{1}0_b = 0.001000\,\underbrace{11}_{\text{group}}\,0\bar{1}000\bar{1}0_b \\
&= 0.0010010\bar{1}0\bar{1}000\bar{1}0_{CSD} = 2^{-3} + 2^{-6} - 2^{-8} - 2^{-10} - 2^{-14}
\end{aligned}$$

Number of reducations in add is two

$$\begin{aligned}
M &= 891 = 11011110\,\underbrace{11}_{\text{group}}\,{}_b = 110\underbrace{11111}_{\text{group}}0\bar{1}_b = \underbrace{111}_{\text{group}}\,0000\bar{1}0\bar{1}_b \\
&= 100\bar{1}0000\bar{1}0\bar{1}_{CSD} = 2^{10} - 2^7 - 2^{-2} - 2^0 = 1{,}024 - 128 - 4 - 1 = 891
\end{aligned}$$

Number of reductions in add is four

Figure 5—For any integer or fractional multiplier M, the CSD conversion of multipliers is done to reduce overhead. Stepwise grouping of adjacent binary 1s with the number of reductions in each case is shown. The integrity of the multiplier after conversion to CSD is also verified.

INTEGER MULTIPLIER

Horner's method is easily extended to integer multipliers using the same concept. The only difference is in the search for ones. The search is now from the leftmost bit to

Methods	CPU cycles	Code size	Result	Absolute error
Horner's method	33	68 bytes	10656	0.38979
Horner with CSD	27	56 bytes	10656	0.38979
Existing method (14)	107	54 bytes	9954	702.38979
Existing method (15)	107	54 bytes	10665	8.61021
C Floating-point library	427	322 bytes	10656.38979	0

Table 1—This evaluation of Horner's method is done with comparisons to other methods on the MSP430 platform. For consistency, the same multipliers are chosen and the result for each multiply is shown. Cycle count directly relates to the overhead on the CPU and absolute error indicates the accuracy of each method.

the rightmost bit before the binary point. The multiplier M in this case is any integer. Additional care must be taken to ensure that the result of the multiply does not exceed the range for representation in the microcontroller. The second example shows a similar scheme when the multiplier is an integer type. Only design equations have been provided for this multiplier with x taking any form (integer/fraction). Consider the multiplier $M = 77 = 1001101_b$. The following are design equations:

$$x \times 2^3 + x = x_1 \qquad [9]$$

$$x_1 \times 2^1 + x = x_2 \qquad [10]$$

$$x_2 \times 2^2 + x = x_3 \qquad [11]$$

$$\text{Final product} = x_3 \times 2^0 \qquad [12]$$

The weights in design Equations 9 through 11 are all positive powers of two rather than negative, as the direction is now from left to right, as opposed to right to left for fractional multipliers. The result of back substitution for the integer multiplier is shown in Figure 4.

REAL MULTIPLIER

Procedures for pure fraction and pure integer multipliers were shown separately to distinguish their implementation. Once this is established, a real number multiplication can easily be realized using either of the two approaches. The multiplier is scaled up or scaled down to either pure integers or pure fractions, and Horner's method applied to them. Once the multiplication is complete, the result must be scaled accordingly. The resulting error is extremely small, similar to Equation 1. This makes filtering an easy task without the overhead of a C floating-point library.

CSD REPRESENTATION

The CSD format or representation is tailor-made for algorithms that rely on bits with a value of 1 for their design.[3] The CSD format for its representation uses a ternary set {–1, 0, 1} compared to a binary set {0, 1}. If looked at carefully, the number of steps in Horner's algorithm depends on the number of 1s present in the multiplier. The CSD format attempts to reduce the steps by grouping consecutive 1s in the multiplier and replacing them with a combination of the ternary set {–1, 0, 1}. This modification reduces the number of add operations for multipliers that have groups of consecutive 1s. By examination, the CSD representation would never have adjacent 1s or –1s.

Don't be confused by the term "ternary set" because the process of introducing –1s in a number is merely to replace some of the additions by subtractions. The grouping is done starting from the rightmost bit and proceeding left. Figure 5 shows an example with the steps involved in binary to CSD conversion and subsequent reduction in add operations. The ternary element –1 is represented as $\bar{1}$ in Figure 5.

For the fractional multiplier, starting from the rightmost, replace the first group of 1s (2^{-14} to 2^{-11}) by a combination of {–1, 0, 1}. The four 1s are combined and $\bar{1}$ is placed at the rightmost bit position (2^{-14}), zeros at the remaining position (2^{-13} to 2^{-11}), and a 1 in the 1 bit position to the left of this group (2^{-10}). This procedure is repeated for subsequent groups of ones already present or created from previous groupings. A similar procedure is shown for the integer multiplier. The reduction in add operations is two in the fractional case and four in the integer case. This reduction is multiplier-dependent and effective only when there are a number of groups of consecutive

Listing 1—This is a section of typical MSP430 microcontroller code for an integer-real multiply. The dedicated code mainly consists of add/subtract and shift instructions only. For an MSP430 microcontroller, the add/subtract and shift instructions are single-cycle.

```
inv.w R13            ; 2's compliment since the last digit is -1
add.w #1,R13
rra.w R13
rra.w R13
rra.w R13
add.w R12,R13        ; X1=-X*2^-3+X
rra.w R13
rra.w R13
sub.w R12,R13        ; X2=X1*2^-2-X
rra.w R13
rra.w R13
rra.w R13
rra.w R13
rra.w R13
rra.w R13
sub.w R12,R13        ; X3=X2*2^-6-X
rra.w R13
rra.w R13
rra.w R13
rra.w R13
add.w R12,R13        ; X4=X3*2^-4+X
rla.w R13            ; Upscale final result by 16
rla.w R13
rla.w R13
rla.w R13            ; Final result of multiplication
mov.w R13, R12       ; Value returned to calling function
ret
END
```

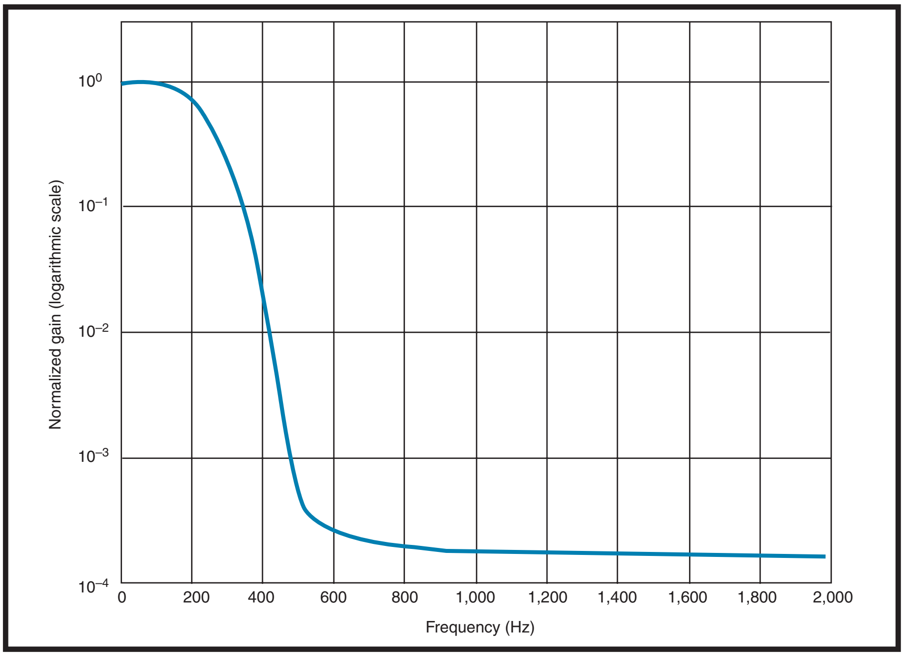


Figure 6—Horner's method is used to implement a low-pass filter with a cut-off of 300 Hz. The response is depicted as a gain versus frequency plot and is shown to conform to the design.

ones in the multiplier. The CSD format produces the same results for multiplies or divides with a slight reduction in CPU overhead. The introduction of the CSD format introduces a small change in the way Horner's algorithm is implemented. Because there would be a ternary set, all –1s would be subtracts and all 1s would continue to remain adds. This subtract, when necessary, would merely replace the add operation implemented at the end of each step.

IMPLEMENTATION ON THE MSP430

Horner's method works most efficiently on microcontrollers that support single-cycle add and shift operations. For example, the MSP430 CPU supports this requirement and also performs single-cycle register-register move operations, enabling fast multiply. A section of the MSP430 microcontroller's assembly code is shown in Listing 1. Table 1 shows a comparison of the Horner's method, Horner's method using CSD, existing algorithms, and the floating-point C library. Performance measures such as code size, CPU cycles, and the final result have been shown for a standard integer-real multiply. Results for filter implementation using Horner's method using CSD are implemented. An approximate frequency response using a simple square and add operation in the filtered signal across various frequencies is shown in Figure 6 and Figure 7.

Table 1 shows the comparison of Horner's method and its variation to alternate methods for an integer-real multiply of the number 711 and the real multiplier 14.98789. The number 14.98789 is first scaled down by 16 (right shift by four) to get 0.936743125. The result is then scaled up by the same number 16 (left shift by four). The result of Horner's method is the closest to the floating-point library implementation losing only the fractional part of the final result. This result is as good as it can get considering its implementation is on a fixed-point processor. The existing method, although generic in implementation, loses precision due to finite-word length effects at the start of the multiply. To be fair to the integer-integer multiply algorithm, both cases of rounding up to 15 and rounding down to 14 are shown. The multiplier, even if rounded-up, has significant error (see Table 1).

Although Horner's method requires dedicated code for each multiplier, the accuracy it provides is too tempting to use it. The CSD format is just an additional option to further reduce the execution cycles and code size.

For real-time filtering operations, Horner's method is certainly preferred over the other method because it is at least three times faster than the existing methods. These methods have limited accuracy with the option of scaling and using 32-bit registers to improve accuracy. The floating-point library is almost not an option because it is almost 15 times slower. To exhibit the performance of filtering using Horner's method, two filter implementations have been shown. The first example is a FIR low-pass filter and the second is a FIR band-pass filter. In both cases, the normalized gain in dB versus frequency is shown. Limited frequency resolution has been chosen because the intent is to show the filter's performance for a pre-fixed frequency sweep. In a real-world application, frequency resolution does not have a role to play. The performance depends on the number of bits chosen for the coefficients and the filter order. For every increase in order, the increase in the number of CPU cycles can vary between 30 and 35 CPU cycles, which also includes memory updates. These are huge savings when compared to a floating-point library implementation.

Figure 6 shows the response of a thirtieth-order low-pass filter with its cutoff set at 300 Hz at a sampling frequency of 4,000 Hz. The entire filter takes 1,030 CPU cycles with 31 integer-real

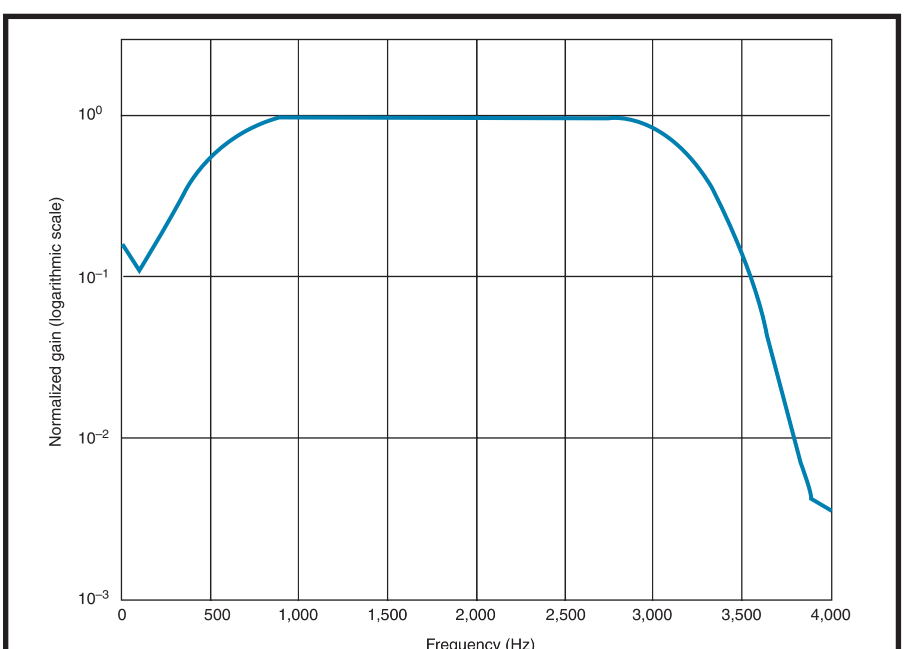


Figure 7—When Horner's method is used to implement a band-pass filter to extract the voice band of 300 to 3,400 Hz, the response is depicted as a gain versus frequency plot and it is shown to conform to the design.

multiplies and 30 memory-memory moves at each stage and occupying a code size of 1,696 bytes. The MSP430 microcontroller supports a real-time operation for this case. If a floating-point library was used instead, the cycle count would be a minimum of 14,000 cycles preventing any real-time filtering.

Figure 7 shows an implementation of a twentieth-order band-pass filter specifically intended to limit the frequencies to the voice band, ideal for speech applications. The passband is set from 300 to 3,400 Hz with a sampling frequency set at 8,000 Hz. The total instruction cycles in this case is just 677 CPU cycles with a code size of 1,110 bytes. Both these filters are very close to their respective floating-point implementations.

ACHIEVE EFFICIENCY

This article focused on efficient multiplication and division of microcontrollers without a hardware multiplier module. The design steps to implement Horner's algorithm were shown with examples. The CSD format was introduced and its effects on reduction of steps was portrayed. This method not only showed superior performance, but also removed the fear of CPU overhead on the real-time implementation of digital filters. The number of CPU cycles and code size depend entirely on the resolution chosen for the filter coefficients. There cannot be a compromise on the integer part, but fewer resolutions for the fractional part can significantly reduce the CPU cycles with a compromise to performance. This method is certainly not limited to processors without a hardware multiplier. Instead, it paves the way for integer-real multiply on fixed-point machines. With memory getting cheaper by the day, code size can never pose a limitation for such a powerful scheme!

Kripasagar Venkat (kripa@ti.com) is an applications engineer for MSP430 microcontrollers at Texas Instruments. He holds a Master's degree from the University of Texas at Dallas with an emphasis on digital signal processing. His current interests include signal processing and filter design for low-power applications.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2008/212.

REFERENCES

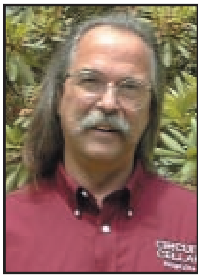
[1] S. K. Mitra, *Digital Signal Processing: A Computer-Based Approach*, McGraw-Hill, New York, NY, 2001.

[2] C. Hamacher, Z. Vranesic, and S. Zaky, *Computer Organization*, McGraw-Hill, New York, NY, 1990.

[3] R. M. Hewlitt & E. S. Swartzlantler, "Canonical Signed Digit Representation for FIR Digital Filters," 2000 IEEE Workshop on Signal Processing Systems, Oct. 11 to 13, 2000.

SOURCE

MSP430 Family of microcontrollers
Texas Instruments, Inc.
www.ti.com



by Jeff Bachiochi

Navigating the Heavens

With Celestron's SkyScout, you can explore outer space without leaving your backyard. The handheld device gives you point-and-click convenience to identify thousands of stars, planets, constellations, and deep-space objects. You don't even need instructions. Just turn it on and follow the directions on the LCD. ARM processor power brings the heavens to you.

Adding exterior lighting to streets, parking lots, buildings, and signage makes our nights safer, but it unknowingly adds to the growing light pollution problem that may threaten some life on our planet (see Photo 1). Light pollution isn't obvious unless you like to stargaze. If you do, you quickly realize its effect of washing out the nighttime sky. Civil engineers are now required to design with directional lighting to help minimize this effect. Light pollution interferes with the way many plants and animals function. Although we cannot eliminate light pollution, we can reduce its effects and save energy by using illumination wisely (as we should every resource). Although light pollution interferes with this month's topic, it's not the focus here. However, you are encouraged to learn more about light pollution and its effects on our planet by visiting your local library or using your favorite search engine.

This month, I want to discuss the design of a device that makes it simple for anyone to enjoy stargazing. Celestron has been making telescopes since 1960 and it recently released the SkyScout, a handheld device that gives its users a point-and-click way to identify thousands of stars, planets, constellations, and deep-space objects.

The celestial sighting device uses an exterior LCD to give instructions and information. The device has two basic modes of use, Identify and Locate. Pressing the Target button automatically identifies any celestial object that is centered in the sighting tube. Alternately, you can flip through a database of objects on the LCD, choose one, and the SkyScout will steer you toward it via unique pointers in the sight tube.

To enjoy the SkyScout, you don't need to know anything about celestial coordinates, just turn it on and follow the clear instructions. However, to understand how this device works, you'll need to know a little about the sky.

EARTH AND SUN

The Earth rotates about its North/South Pole axis once a day (24 hours). It also orbits around the sun once in a year while it spins (365 1/4 rotations or days). The Earth's axis is not perpendicular to its orbit but tilted about 23.5°. (As with the mechanics of most objects, the tilt isn't constant. But in an effort to make things less confusing, we'll ignore long-term effects.)

The Earth's tilt is responsible for the change in seasons. At one point during the orbital year, the Earth's North Pole will be closest to the sun (the solstice). The northern hemisphere is midsummer with its longest period of daylight. The southern hemisphere is mid-winter with its shortest period of daylight. Three months later, each pole will be equidistant from the sun (the equinox). While the northern hemisphere heads toward winter with shorter periods of daylight, the southern hemisphere looks toward summer and longer periods of daylight. At the half-year mark, another solstice occurs. Now the South Pole is closest to the sun and the southern hemisphere experiences summer, while the northern hemisphere is midwinter. After another equinox at nine months, the Earth heads back to its original position and the yearly seasonal cycle begins again.

The Earth is geographically mapped using latitude and longitude. Latitude is a north-south measurement. Lines of constant latitude run east-west around the Earth's circumference, parallel to the equator. Longitude is an east-west measurement, with lines that run pole-to-pole dividing the Earth into "orange wedges." The prime meridian is the longitudinal arc between the two poles that passes through Greenwich, England. It has been assigned 0° (with 180° on the opposite side of the Earth). Longitudinal lines to the east of the prime meridian are designated 0° to 180°E. To the west of the prime meridian they are designated 0° to 180°W. Latitudinal lines are north/south measurements based on the equator. The equator is designated as 0°. From the Earth's center, angles measured from the equator (0°) to either pole (90°) divide each hemisphere into rings parallel with the equator, either 0° to 90°N (north of the equator) or 0° to 90°S (south of the equator). For instance, the location 0° longitude and 0° latitude defines a point off the shore of Africa.

It wasn't until 1884 that Greenwich

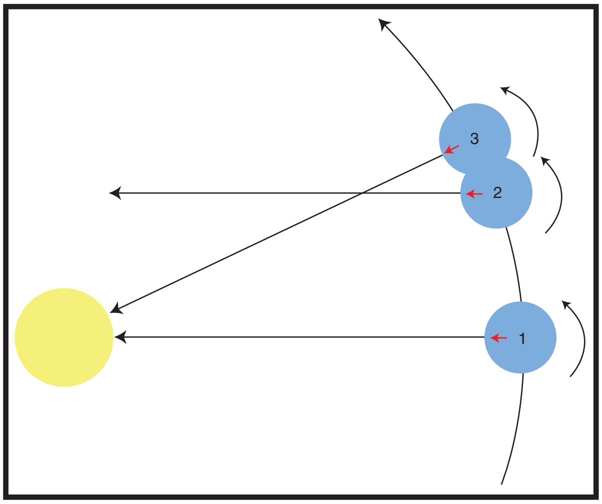


Figure 1—This graphic demonstrates a solar day and a sidereal day. A sidereal day is the difference between time 1 and time 2, or 23:56:04 (based on the change in the Earth's position to a distant star). A solar day is the difference between time 1 and time 3, or 24:00:00, an additional 00:03:56 (based on the Earth's change in position to the sun). On a prograde planet like the Earth, a sidereal day is shorter than a solar day. At time 1, the sun and a certain distant star are both overhead. At time 2, the planet has rotated 360° and the distant star is overhead again, but the sun is not (1→2 = one sidereal day). It is not until a little later, at time 3, that the sun is overhead again (1→3 = one solar day). (Source: http://en.wikipedia.org/wiki/Earth_rotation)

was adopted as the universal prime meridian (0°). Imagine the confusion with mapmakers all using their own designation for the prime meridian!

EARTH AND THE COSMOS

We are so enamored with the sun that we sometimes fail to look at the bigger picture. Our solar system, the sun, and all nine planets (yes, I still consider Pluto a planet), are but a speck in a spiral arm of the Milky Way galaxy. For a little comparison in scale, let's use the speed of light (the fastest measure we know). From the sun, it takes about 8 min. for light to reach Earth (about 80 min. to reach Pluto). This makes the diameter of our solar system about 160 light minutes, or 2.6 light hours (about 1/10 of a light day or 1/3,650 of a light year). The size of the Milky Way is about 65 million times larger, or more than 18,000 light years! It contains hundreds of millions of stars.

It is enough to know that the Milky Way galaxy is only a speck in the universe. The number of galaxies in the universe is almost beyond comprehension. All that is out there is in constant motion. To the average Joe on Earth, what he can see from Earth (by the unaided eye) defines his playing field. From our vantage point, all objects outside our solar system appear to be motionless. If the first stargazers were able to take photographs of the constellations thousands of years ago and we compared them to what we see today, there would be differences. That is because most stars in a constellation are at different distances from us. Each moves at its own rate, changing the general shape of the constellation's pattern by very small amounts.

Imagine for now that we had no sun and it was night 24 hours a day. This means that we would always be able to see the stars. In 24 hours, we would see a complete tour of the constellations as the Earth makes one complete rotation. If we picked out a star, such as Betelgeuse (in the constellation Orion), and timed how long it takes the Earth to rotate bringing Betelgeuse back to the same position in the sky, it would not be 24 hours (24:00:00), but 23:56:04. Hmm.

Because the rotating Earth moves in an orbit around the sun, it needs to rotate past 360° to get pointing back to the sun (see Figure 1). So, when we say the sun rotates 360° every day, this is in reference to the sun (a minor player). This definition of a solar day is 24:00:00 in length. From the perspective of the cosmos, what we take for granted is all wrong. The celestial or sidereal day is 00:03:56 shorter.

With the sun back in place, things don't really change. It's just our perception that changes because we base our lives on the solar day and not the sidereal day. With the sun back in the picture, we see the stars only at night, and our view of them changes by 00:03:56 each night. It takes a full year to see all that the heavens has to offer because we get to "see" the stars only when we face away from the sun.

The mapping system of the heavens is similar to our geographical coordinate system on Earth. Imagine a clear sphere surrounding the Earth that is able to stay fixed with the universe so the Earth rotates within it. The Earth's axis extends through the sphere, effectively giving this new sphere poles. The sphere has longitudinal wedges that are marked by time (like time zones). Whereas the geographical coordinate system uses the prime meridian's intersection with the equator as the starting reference point, the celestial coordinate system uses the arc between its poles that passes through the first star in the constellation Pisces as a reference of time zero (00:00:00). Twenty-four wedges create a longitudinal line every 15 degrees, or one hour (360°/24 hours) (see Figure 2). Thus, time is used to indicate a "right ascension" to the west in hours, minutes, and seconds (the amount of time it takes the Earth to rotate from the reference to the object of desire, 00:00:00 to 24:00:00).

Just like latitude lines in the geographic coordinate system, declination (DEC) is given in degrees north and south of the equator. Only in the celestial coordinate system, the degrees are designated by using (+) instead of (N) and (–) instead of (S).

I like to think of the radius of this

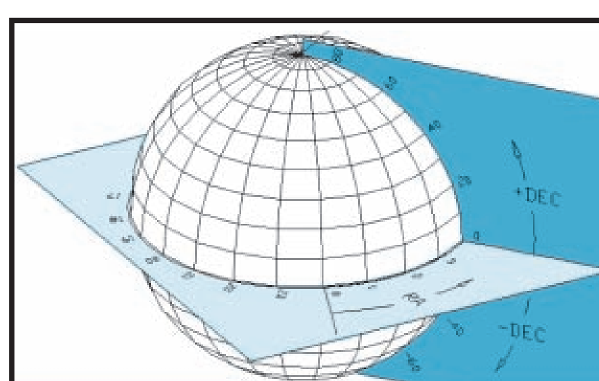


Figure 2—The celestial coordinate system uses time designated as right ascension (RA) and angle of declination (DEC) from a reference point where the celestial equator crosses the arc passing through the first star of the constellation Pisces. (Source: www.celestron.com/c2/images/files/downloads/CPC800_11073.pdf)

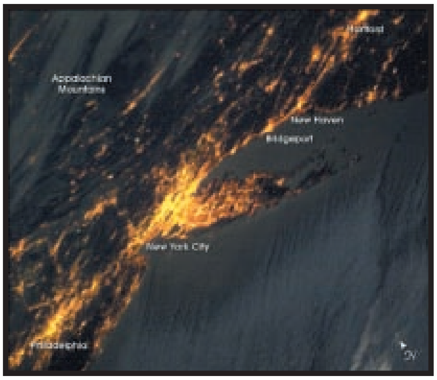


Photo 1—You can clearly see the light pollution of the eastern seaboard as seen from the space shuttle as it passes over the Long Island Sound. (Source: Earth Sciences and Image Analysis Laboratory, NASA, Johnson Space Center)

clear sphere to be variable and just smaller than any object of interest. This way, any parallax error (the difference in viewing angle between the center of the Earth—the center of the sphere—and a viewer on the Earth's surface) will be insignificant.

The path the stars seem to take (while you spin on the Earth beneath them) will change based on where you view them from. If you lie down at the equator and view the movement of the sky over the period of the night, the stars will rise, move overhead, and set. If you fly to one of the poles and spend the night watching the stars, they will rotate in a circular pattern about the pole.

SkyScout

The small, lightweight SkyScout can be used at length without fatigue. Any product's success will depend on many factors, but the user interface is the most important. If the product is intuitive, it is more likely to be successful. The SkyScout has a friendly appearance with all controls, except for one, mounted on one face with a backlit (red to preserve night vision) LCD (see Photo 2). Each of the nine large rubber buttons (with a construction similar to a remote control) is clearly marked with its function (i.e., Locate). A Target button is positioned for easy access beneath the forefinger while viewing an object through the sighting tube. Unlike a telescope or binoculars, there is no magnification through the sighting tube. When you look at an object through the SkyScout's sighting tube, the Target button provides a snapshot of the device's position. This is used for identifying objects. In Locate mode, a database of objects is displayed via the LCD. Once an object is selected, eight LEDs around the inside of the sighting tube direct you to move toward the object until it is directly in the center of the sighting tube. The simplicity of coaxing you to move using a ring of directional LEDs disguises the complicated sensors and math computations associated with the action. This is the basis of good product design.

Cell phones, navigation aids, the LoJack, and other products take advantage of the system of satellites in geosynchronous orbit to gather positional information. The SkyScout uses an internal GPS module to calculate the precise position of its user. It requires sufficient signal from three satellites to triangulate the user's position on Earth (longitude, latitude, and elevation). It may attempt to receive signals from up to 12 satellites. Once the SkyScout has acquired a fix, you

Photo 2—An LCD and controls are conveniently located on the side of the SkyScout. Two AA batteries supply power for hours of nighttime viewing. (Source: www.myskyscout.com/skyscout_tour/index.html)

Name	Polaris
Common name	The North Star
Constellation	Ursa Minor
Bayer	1-Alpha Ursae Minoris
Distance	431.42 LY
SAO	308
HIP	11,767
Spectral	F7;Ib-IIv SB
Apparent magnitude	1.97
Absolute magnitude	–4.60
RA	2 h 31.84 min.
DEC	89.2640°
Type	Binary star
Separation	18.4000 arc sec

Table 1—SkyScout's database reveals plenty of information about constellations, including common and proper names as well as brightness and where it is located in the celestial coordinate system.

Name	Jupiter
Diameter	139,822 km (86,833.7 miles)
Mass	1.90e + 27 (318 Earths)
Orbital period	11.86 Earth years
Rotation period	9.94 Earth hours
Orbit radius	1,557 million km
Apparent magnitude	–1.85
RA	17 h 19.96 min.
DEC	–22.8579°

Table 2—The planetary database includes technical information, such as size and mass in addition to the planets' orbital physics.

can review the identified satellites and their associated strength along with your present calculated position. GPS technology is the first step in making this product user-friendly. It eliminates the need for a user to determine longitude, latitude, and time of day and then have to enter them manually.

The remaining sensor technology (physical position of the SkyScout) is used to locate or identify objects in the sky. The order of the next two procedures depends on whether you are locating or identifying. When using the locate function, the database is used to determine the celestial position of the object. Then the positioning function directs the SkyScout toward the object's position relative to the user's position. When the identify function is chosen, the positioning functions determine the celestial position of the viewing tube and then check the database for an object that matches that celestial position.

DATABASE OF OBJECTS

Objects outside of our solar system require a minimum of data. These objects can be considered standing still in reference to our lifetime. While only RA and DEC are necessary to fix its position on the celestial sphere, other data provides plenty of educational fodder. The scientific data displayed for Polaris is shown in Table 1.

Objects within the solar system have a life of their own. Because they are (relatively) close by and have periodic orbits like that of Earth, their positions relative to the Earth are always changing. Here a fixed RA and DEC won't work. The RA and DEC must be calculated. This is based on where the Earth is within its orbit and where Jupiter (in this case) is within its orbit. A two-line element (TLE) is a data record that describes an orbit's features. Date and time information from the GPS enables the SkyScout to calculate orbit positions based on a TLE. The two calculated orbital positions (in this case Earth and Jupiter) can be used to determine an RA and DEC for locating Jupiter. The scientific data displayed for Jupiter is shown in Table 2.

GPS

The Global Positioning System (GPS) is a network of satellites, set in geostationary orbit around the Earth, which transmits a time-synchronized signal. The National Marine Electronics Association (NMEA) developed the specification that defines the data issued by GPS receivers. One such sentence (ASCII data) is shown below:

```
$GPRMC,123519,A,4807.038,N,01
131.000,E,022.4,084.4,230394,
003.1,W*6A
```

"RMC" is the recommended minimum sentence C data. "123519" is the time of the Fix, 12:35:19 UTC. "A" is the Status (A = active or V = void). "4807.038,N" is the latitude 48° 07.038′N. "01131.000,E" is the longitude 11° 31.000′E. "022.4" is the Speed over the ground in knots. "084.4" is the Track angle in degrees True. "230394" is the Date, 23rd March 1994. "003.1,W" is the Magnetic Variation. "*6A" is the checksum data, which always begins with an asterisk.

You might want to visit the NMEA web site or read one of my a previous columns for more information on using GPS data ("Where's Waldo?: Pinpointing Location by Interfacing with a GPS Receiver," *Circuit Cellar* 126, 2001). The more satellites a GPS receiver has to work with, the more accurately it can calculate its location. You can see from the selected sentence that date and time information is also provided by the GPS.

The SkyScout uses the GPS to locate the user on the surface of the Earth (where), and also to define a snapshot in time (when). This information is used to define a point in time when the user geographic coordinate system is referenced to the celestial coordinate system, as if the Earth has stopped rotating in a known position in reference to all of the objects in the sky.

WHICH WAY IS UP?

At this point, all we've done is stop the Earth. (We've taken a snapshot in time.) We still need to know where to look to find an object or where we are looking to identify an object. The SkyScout uses magnetic and accelerometer sensors to determine its orientation based on the Earth's magnetic field and gravity.

Several months ago, I explained how to use an accelerometer to determine slope ("What's the Slope?: Use an Accelerometer to Measure Slope," *Circuit Cellar* 202, 2007). The Earth's gravity looks like a 1-G acceleration toward the center of the planet. Once the GPS has stopped and the Earth and the sky are in a fixed position, the accelerometer will provide a reference vector from the center of the Earth, through your position, and extend directly overhead into the universe. This vector enables the

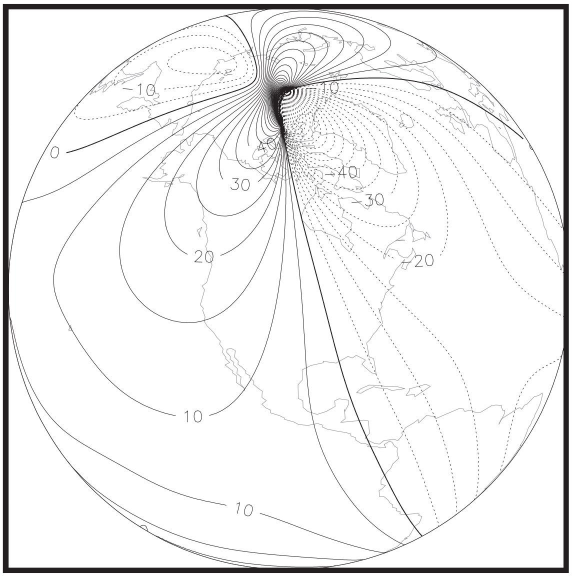


Figure 3—This map shows the declination offsets for the world as of the year 2000. It is interesting to note that the magnetic poles are not 180° opposite one another as are the true North and South poles (the Earth's axis of rotation). (Source: http://geomag.usgs.gov/charts/IGRF2000.dec.na.pdf)

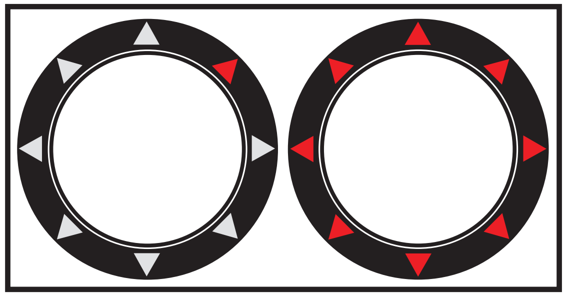


Figure 4—Red directional LEDs prompt you to move the sighting tube in the direction necessary to bring the object of interest into view. All LEDs on means you are there. (Source: www.celestron.com/c2/images/files/downloads/1157656360_skyscoutmanualw.pdf)

SkyScout to point directly toward the point in the celestial sphere that the GPS's data has been able to calculate.

Remember that the geographical coordinate and celestial coordinate systems have the N/S pole axis in common. To move to other locations on the celestial sphere, we need to know which direction is north. The SkyScout uses sensitive magnetic sensors to accomplish this. (The SkyScout is so sensitive to magnetic fields that it provides mu-metal shields to block any magnetic field interference from the AA batteries.) Most can tell you that a compass needle will point north. Some know that this is not really north. Few know where on Earth it actually points because it is constantly in motion. Presently, it is located in northern Canada and it is moving northwest by miles a year. (It is interesting to note that charged particles from the sun cause it to wander in a daily elliptical perturbance.) Mapmakers note the difference between true north and magnetic north for a given map by declaring the magnetic declination for the area. As you pass from the East to the West Coast in the U.S., this declination changes from about –20° to 20° (see Figure 3). You can see that the declination is important to accuracy and its value is determined by where on Earth you are located. You may have noticed that the GPS output string includes a magnetic variation element.

Now that we know which way is north, we can move to any other coordinate in the celestial coordinate system from the known point directly overhead. But how does the SkyScout lead the user toward an object of interest? The sighting tube has eight LEDs located around its interior circumference. Real-time calculations, based on the celestial map and position sensors (up and north), determine the shortest route and guide your movement toward the object of interest. One or more of the LEDs are illuminated to gently direct the

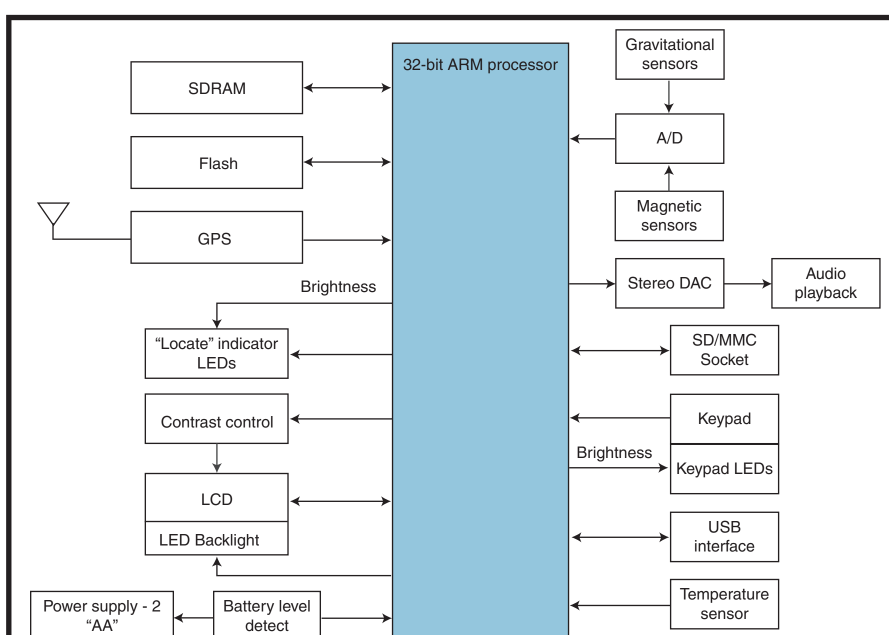


Figure 5—This block diagram shows what peripherals support the ARM processor's bid to make one of the most user-friendly, high-tech, educational instruments to be developed in recent years. The SkyScout rivals the iPhone for cool.

user toward the destination (see Figure 4). Because the SkyScout senses magnetism and acceleration in all three axes, the processor can continue to calculate the correct direction independently of how you are holding the SkyScout. Object identification and location will work night or day and even through the Earth, as if the sun and Earth don't exist. Although we can't "see" objects during daylight or on the opposite side of the Earth, the fact that the SkyScout points there reinforces the fact that just because we can't see it doesn't mean it's not there.

To provide the SkyScout with the necessary computational power, the design was built around Samsung's S3C24310, an ARM9 series 32-bit processor. The device supports plenty of goodies, including USB, LCD, and SD interfacing (see Figure 5). The total design is based on three PCBs, GPS, an LED directional ring, and the main processor board. Photo 3 shows the LED directional ring (removed from its slot in the sight tube) and the GPS board (with the GPS patch antenna showing). The main board holds most of the other components with the opposite side saved for the LCD and membrane push buttons (see Photo 4). A USB and headphone jack can be seen on the left and an SD card slot to the right.

EMBEDDED EDUCATIONAL

The SkyScout's ARM processor puts its power to good use in providing real-time calculations as its sensor arrays indicate a continually changing position. But its expertise doesn't stop there. Integrated SD card slot and stereo DAC interfaces add to the design features. The SkyScout's internal field guide provides

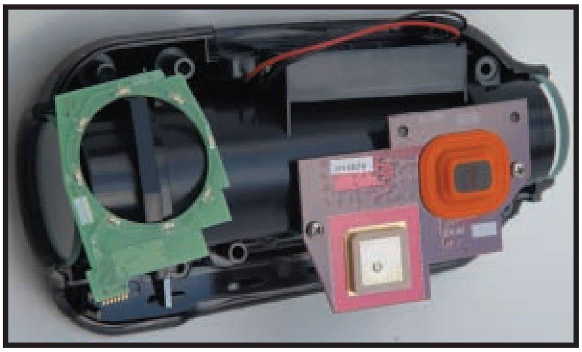


Photo 3—The LED direction ring has been removed from the sighting tube slot on the left. The GPS module is mounted beneath the right PCB. The GPS patch antenna can be seen on the topside of the PCB along with the membrane Target button.

text of astronomical facts (about astronomers, man-made objects, comets, and asteroids) and a glossary. The guide includes a six-part audio guide to astronomy and narrations or text descriptions and scientific data of approximately 200 objects. The SD card interface permits expanding its knowledge bank.

One of the typical improvements to newer embedded designs is the ability to update the operation system without major surgery. The processor's integrated slave USB port supports updates to the SkyScout.

FRIENDLY SKIES

Except for picking out "the big dipper," most of us must plead ignorance to much of what's out there. While location information of celestial objects is well documented, it requires a clock and calendar (time and date), a map of the world (location), a compass (direction of north), and a protractor (for azimuth) to calculate where and when to look. This complexity scares off most people who might otherwise enjoy knowing more about the night sky. Our success as designers depends on our ability to take

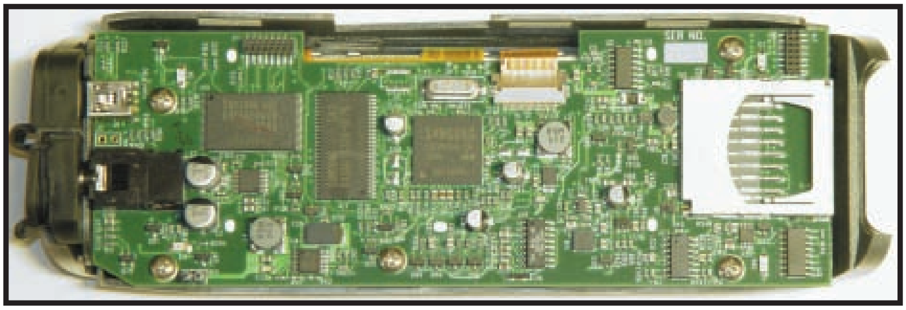


Photo 4—The main PCB contains most of the electronics including the Samsung 32-bit processor, the magnetic and accelerometer sensors, and all of the support circuitry.

sensor technology and provide a simple solution to a complicated problem. The SkyScout is a perfect example of how individual advances can be combined to provide a solution that was once unthinkable.

While combining the proper electronic components in the right packaging provides a solution, a convoluted user interface can ruin an otherwise great product. The ultimate interface requires no "user's manual." As designers, we need to pay as much attention to the basic concepts of the design as we do to coding the algorithms within. Can you say, "iPhone"?

Author's note: Special thanks to Mike Lemp, the architect and father of the SkyScout.

Jeff Bachiochi (pronounced BAH-key-AH-key) has been writing for Circuit Cellar *since 1988. His background includes product design and manufacturing. He may be reached through the magazine (jeff.bachiochi@circuitcellar.com) or his web site(www.imaginethatnow.com).*

RESOURCES

J. Bachiochi, "Car 54, Where (Exactly) Are You?: Adding E-Mail Capabilities to Your Project," *Circuit Cellar* 127, 2001.

———, "Magneto-Inductive Direction Finding," *Circuit Cellar* 80, 1997.

———, "What's the Slope?: Use an Accelerometer to Measure Slope," *Circuit Cellar* 202, 2007.

———, "Where's Waldo?: Pinpointing Location by Interfacing with a GPS Receiver," *Circuit Cellar* 126, 2001.

National Marine Electronics Association (NMEA), www.nmea.org.

SOURCES

SkyScout
Celestron
www.celestron.com

S3C24310 Evaluation board
Samsung
www.samsung.com

by George Martin

Do You Want to Do a Design?

Linked Lists

After months of learning C, it's time to apply what you've learned in a real-world application. George explains how to tackle a real design problem from start to finish.

For more about C language and writing code, visit George Martin's blog, CCI C Programming Design Review, at http://ccidesignreview.wordpress.com/. Code and additional files are also available on the *Circuit Cellar* FTP site.

You've been learning C for a while now. Hopefully, you can read and write C code. This month, I would like to take a real design problem from the beginning to the end. Along the way we'll create a design, make design trade-offs, accept changes, and code the design. This should be interesting.

DESIGN CHALLENGE

A few months ago, a customer asked if I would add a feature to their product. The customer wanted me to have their product call out using the telephone interface whenever a problem was detected. The design already had a PBX interface that passed FCC Part 68. The PBX interface was used to permit a user to call the unit and enter commands. After a few more discussions, the customer thought that there might be 10 events that would cause a call out to be initiated and that there should be four telephone numbers to dial for each of the 10 events. The numbers would be dialed and a message played to the user. If the user entered the proper response, the error event would be considered reported; if not, the next number would be called.

This is a straightforward design and coding task, but some issues need to be looked at. First, the hardware interface does not have call progress monitoring. That means that we can not detect activities or conditions such as dial tone, ringing, or call answered. I suggest that we use timers in a state machine design to pace the code through the calling process. We can assume certain times for each of the states of the telephone line interface. Because we're calling friendly users (their numbers have been determined to be available for calling), this should be acceptable. The detail of interfacing with the PBX hardware is very dependent on the specific interface. With your indulgence, I would like to skip over this part of the design. Your interface will most likely be different than mine, and you'll need to match your software design to your calling hardware (see Listing 1).

Also, we need a way to enter the numbers to dial for each of the problem types that we can detect. And I would add a log file to log the activity. I would like to have a record that we tried to call and no one answered or we could not get a free line.

DATABASE DESIGN

The internal database would look like this:

```
INT8 DialOut[MAX_GROUPS][MAX_NUMB]
[MAX_DIGITS];
```

Database design is really straightforward. It's just one large array of 8-bit entities indexed by groups (type of error detected) and the digits in the telephone number. `MAX_GROUPS` is defined as 10, `MAX_NUMB` as 4, and I'm going to define `MAX_DIGITS` as 25. That should be large enough to dial numbers like 9,,,18005551212,,123 (20 INT8s). Where the "," indicates a pause. So, in my example we're dialing 9, pausing three units, dialing a 1-800 number, then pausing two units, and then dialing extension 123. I think that should cover just about any dialing issue that might come up.

This design uses `MAX_GROUPS` × `MAX_NUMB` × `MAX_DIGITS` = 10 × 4 × 25 = 1,000 bytes of memory. That data needs to be saved in a nonvolatile memory area so that it's preserved between power outages. That nonvolatile type of memory is usually EEPROM and it is more expensive and smaller than RAM or EPROM. So, we need to use it wisely. I think this design meets those requirements.

Great, I wrote up this design approach and was ready to start. Not so fast. The customer called and asked if we could also handle 30 events with only one number for each event. Well, of course we could, but how? If we extend the present design, `MAX_GROUPS` becomes 30 and `MAX_GROUPS` × `MAX_NUMB` × `MAX_DIGITS` = 30 × 4 × 25 = 3,000 bytes of memory. In the second design request, we would be using one of the four available numbers (25%), and in the original design we would be using 10 of the 30 groups (33%). Neither is using these valuable resources very well and I bet the customer will come up with alternative requests that lead to ever more inefficient EEPROM usage. This is a perfect opportunity to try linked lists.

USE A LINKED LIST

Wikipedia describes a linked list as "one of the fundamental data structures,

Listing 1—This is pseudocode of the problem of dialing out.

```
Detect a problem.
   If the problem has not been reported and acknowledged
      Try to get Outgoing Line
      If Successful
         Dial 1st Number
         Wait for connection
         Send Message
         Wait for Acknowledge
         If Acknowledge is received
            Mark problem as reported and acknowledged
         Else
            Dial 2nd number
            Wait for connection
            Send Message
            Wait for Acknowledge
            If Acknowledge is received
               Mark problem as reported and acknowledged
            Else
               Dial 3rd number
               Wait for connection
               Send Message
               Wait for Acknowledge
               If Acknowledge is received
                  Mark problem as reported and acknowledged
               Else
                  Dial 4th number
                  Wait for connection
                  Send Message
                  Wait for Acknowledge
                  If Acknowledge is received
                     Mark problem as reported and acknowledged
      If not acknowledged Repeat the above process N times
   Once problem is acknowledged mark it as reported.
When Problem goes away clear the reported flag.
```

and can be used to implement other data structures. It consists of a sequence of nodes, each containing arbitrary data fields and one or two references ('links') pointing to the next and/or previous nodes."[1] Stanford University has a great tutorial on the subject and a good set of problems with C code solutions. Refer to the Resources section of this article for more information.

We're going to use a single linked list. Each node will have a number to dial and a linkage pointing to the next number. You will typically see `malloc()` used in Linked list C examples for claiming memory for the lists. Because we're dealing with an embedded system, we're not going to use `malloc()` to allocate memory for our usage. We're just going to assign a sufficiently large amount of memory to hold our data structures. If we need more in the future, it's just one #define to change and recompile. And remember that this memory is EEPROM and it's valuable, so we're trying to use it wisely.

First, we'll build an array for each of the call out groups that contain the first number to try to call. If no number is entered, we'll use a –1 as an indication of that condition. Because we have only 10 (from the first statement of the problem) or 30 (from the second request) possible numbers to call, we can use 8-bit entities for this array. Table 1 shows the array contents.

```
INT8 CallOutGroup[MAX_GROUPS];
```

Next, we'll build a structure containing the digits to dial and a linkage to the next possible number (in this structure) to try to dial. And again a –1 will indicate we've come to the end of the numbers to try (see Table 2).

```
Struct CALL_OUT_NUMBERS { // define
// the structure
  INT8 DialNumb[MAX_DIGITS];
  INT8 NextNumb;
};
struct CallOutNumbs[MAX_NUMBERS];
// reserve memory
```

Look at Tables 1 and 2. And remember C arrays start at 0. The first entry `DialNumber[0]` is empty (–1). The second number 1-800-555-1212 is located at `DialNumber[1]`. If that number is dialed and we get no response, we should try the number `DialNumber[2]`. If that number (1-800-555-1111) is dialed and we get no response, we should dial the number `DialNumber[3]`. If that number (1-800-555-2222) is dialed and we get no response, that is the end of the sequence of numbers and we should start the calling out from the first number.

Hopefully, you can see that an event starts the process by looking up the first number using the `CallOutGroup[]` array. From then on, the `CallOutNumbs[]` structure defines the number and the next number if there is one. This

Call out group	First number to call
0	–1 (empty)
1	1
2	3
3	–1 (empty)
—	—
29	–1 (empty)

Table 1—This is an array of the first numbers to try to call. An entry of –1 indicated an empty number. Remember the array indexing in C starts at 0.

Index	DialNumb [MAX_DIGITS]	NextNumb
0	–1 (empty)	–1 (empty)
1	1.8005551212	2
2	1.8005551111	3
3	1.8005552222	–1 (empty)
4	–1 (empty)	–1 (empty)
—	—	—
39	–1 (empty)	–1 (empty)

Table 2—This is the linked list for our call out sequences. –1 indicates an empty entry.

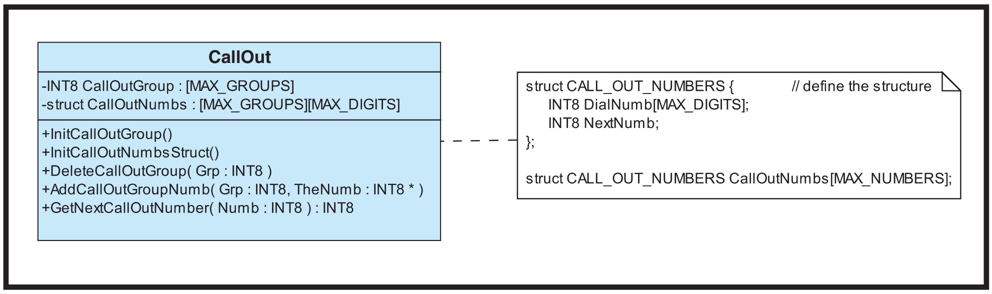


Figure 1—This is the UML case diagram for the initial design of the call out code.

is a classic forward-linked list. And you can also see that it is efficient in memory usage. More callout events linearly increase the size of the `CallOutGroup[]` array. And more numbers to call linearly increase the size of the `CallOutNumbs[]` structure.

DESIGN IMPLEMENTATION

So we now have a design. Let's look at what we need to implement that design. I'm going to present the routines required using UML diagrams. Refer to the Resources section for *Circuit Cellar* articles about UML. UML has several diagrams to describe a system: class diagrams, use case diagrams, collaboration diagrams, state diagrams, activity diagrams, and implementation diagrams. Let's start with the case diagram of the operations for the call out design (see Figure 1).

I defined a class named `CallOut`. Don't worry, we're not getting into object-oriented programming (not just yet, he says, laughing hysterically). But I would put all the call out code in a file named CallOut.c. The CallOut.c file (class) has two static (think of them as private) variables and they are listed in the block below the name of the class. And the routines or procedures of the class are listed in the next block. I also added a note that you see attached to the class design.

Let's look into the routines as they are defined right now. `InitCallOutGroup();` initializes the CallOutGroup. It takes no parameters and returns no parameters. `InitCallOutNumbsStruct();` initializes the CallOutNumbs structure. It takes no parameters and returns no parameters. `AddCallOutGroupNumb();` appends a number to the end of the call out group. It takes the group number and a pointer to the string of digits to dial. It returns no parameters. `DeleteCallOutGroup();` deletes all of the numbers in a call out group. It takes the group number and returns nothing. `GetNextCallOutNumber()` takes the current call out number and looks to see if there is a next number to dial. If so, it returns that next number. If not, it returns –1.

The question to ask ourselves at this point is this: can we complete the design with these routines? Let's see. We need to initialize the CallOut data elements probably at the factory. Perhaps we'll give the user a means to delete all the numbers. Then we need to delete a particular group, add numbers to a group, and get the numbers in order for a group. Looks like we've got it covered.

So, I sat down at the keyboard and fired up Borland's C++ BuilderX on a PC. I've discussed this program in earlier articles. I created a new console application. I'm going to use a DOS window to create and run a test program to exercise the routines we've just designed. Nothing fancy here. It took me about four hours spread out over two sessions to get the code written and debugged. The file TestLinkList.c holds the `main()` routine and runs the test code. I created a file CalOutReports.c to generate printout type routines. These routines report the data in our internal elements out the DOS window for your viewing. And remember that you can pipe the output into a file for hard copy documentation purposes. I have a copy of that file (report.txt) in the files area. I also created a CCILinkedlist.h header file to hold system constants. So the meat of the solution is contained in the files CalOut.c and CalOut.h. Note that a better name for these files would be CallOut.c and .h. That's part of a review process and more on that later.

If we look into CalOut.c, we'll see the routines contained in the UML class diagram (see Table 3).

Routines	Changes
InitCallOutGroup();	No change
InitCallOutNumbsStruct();	No change
DeleteCallOutGroup();	No change
void FindFreeNumber(void);	New
void AddCallOutGroupNumb(INT8 Grp, INT8 *s) ;	No change
GetNextCallOutNumber();	Deleted
INT8 GetFirstCallOutNumberIndex(INT8 Grp);	New
INT8 GetNextCallOutNumberIndex(INT8 Idx) ;	New

Table 3—Here's a list of the routines in the final version of the code and an indication of what's added, deleted, and changed.

Most of our original design had no change. One is deleted and some are added. What's the story? The real world. As I was implementing the code, I saw problems with the design, easier ways to perform a task, or missing elements.

Look at FindFreeNumber. I need to look through the struct `CallOutNumbs[]` to find a free spot to save a new number to. "Well, of course," you might say, but it escaped us at the initial design point. That one is not a big deal.

Next, look at the missing `GetNextCallOutNumber();`, which is replaced with the two routines INT8 `GetFirstCallOutNumberIndex(INT8 Grp);` and INT8 `GetNextCallOutNumberIndex(INT8 Idx);`. If you consider how the database is designed, we first need to look into the INT8 `CallOutGroup[]` array to see if there is a first number entered or if the group is empty. If there is a number, we can start using the forward linkages to keep finding the next number in the group. So we really needed this change, but it could create other problems in the system. Once again, these are the type of real-world issues

that you come across. Now the next step is to redo the UML diagrams to keep them current with the tested design. I'll leave that up to you.

CODE REVIEW

The design is coded and tested. The test cases and results are in the file LinkListReport.txt. The next step is for a code review. What's a code review? Well, Wikipedia says: "Code review is systematic examination (often as peer review) of computer source code intended to find and fix mistakes overlooked in the initial development phase, improving both the overall quality of software and the developers' skills."[2]

Let's have a code review. I'll start it. What happens if a string, longer than the permissible size, is somehow entered for the number to dial? Should we protect against that event? How?

What happens if a group number less than 0 or greater than the maximum is entered? Should we protect against that event? How?

You get the picture. Do a code review and post your comments on the blog I've created for this design review: CCI C Programming Design Review (http://ccidesignreview.wordpress.com/). Then we'll keep that active and see where it takes us.

Let's go. Don't put it off any longer. Next time, let's look deeper into compiler output. We'll take apart the C compiler generated code. Sounds like fun. See you next time.

George Martin (gmartin@circuitcellar.com) began his career in the aerospace industry in 1969. After five years at a real job, he set out on his own and co-founded a design and manufacturing firm (www.embeddeddesigner.com). His designs typically include servo-motion control, graphical input and output, data acquisition, and remote control systems. George is a charter member of the Ciarcia Design Works Team. He's currently working on a mobile communications system that announces highway info. He is a nationally ranked revolver shooter.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2008/212.

REFERENCES

[1] Wikipedia, "Linked list," 2007, http://en.wikipedia.org/wiki/Linked_list.

[2] Wikipedia, "Code review," 2007, http://en.wikipedia.org/wiki/Code_review.

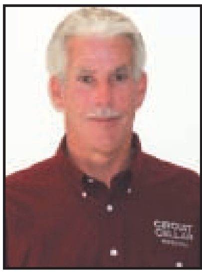
RESOURCES

V. Kosuri, "UML in a Product's Life Cycle," *Circuit Cellar Online* 123, 2000.

M. Manca, "Finite Machines and Microthreads," *Circuit Cellar* 193, 2006.

N. Parlante, "Linked List Basics," Stanford CS Education Library, 2001, http://cslibrary.stanford.edu/103/LinkedListBasics.pdf.

———, "Linked List Problems," Stanford CS Education Library, 2002, http://cslibrary.stanford.edu/105/LinkedListProblems.pdf.



SILICON UPDATE

by Tom Cantrell

More Bits, Less Filling

Tom is a proponent of the 32-bit MCU revolution. This month, he covers several respectable 32-bit chips that deliver performance at a reasonable price. Exciting new applications are on your horizon.

Long-time readers have heard me tell the story of the IC company executive who tried to convince me that the 8-bit market was "dead." The kicker is that he made that statement 25 years (and many billions of 8-bit MCUs) ago.

Indeed, as far as I'm concerned the 8-bit market will still be going strong in another 25 years. True, it may not be as much about new applications anymore. But don't underestimate the fact there are billions of folks around the globe who have a lot of catching up to do gadget-wise. There are plenty of unfilled sockets for 8-bitters.

That being said, I've also been a cheerleader for the 32-bit MCU revolution. These are the chips that deliver performance at a price that enables exciting new applications. Increasingly, software is what makes or breaks a product, and 32-bit chips can tote the code.

Whether high-end or low, the market for MCUs isn't a zero-sum game (see Figure 1). As long as Moore's law keeps delivering more for less, there will be plenty of sockets to go around for everybody.

MCU VS. MPU

The emergence of the latest generation of 32-bit chips once again raises the question of just what's the difference between a microcontroller (MCU) and a microprocessor (MPU)?

In the old days, the answer was pretty simple. If a chip had onboard memory and I/O enabling it to run standalone as a "single chip," it was an MCU. Otherwise (i.e., external memory and I/O), it was an MPU. End of story. As a practical matter, for a long time, you could safely say MCUs were generally "8-bit" and virtually all "32-bit" chips were MPUs. That simple "single-chip" criteria is still pretty valid, but obviously, the business about "bitness" has changed.

Of late, I've adopted a more software-centric approach to the question. These days, I divide the world into two camps. If a chip is designed to run big-ticket software such as Windows, Windows CE, Linux, and more, it's an MPU. If it isn't, it's an MCU.

Obviously, there are gray areas. The dividing line between "big-ticket" and "little-ticket" software can be blurry. For example, the term "RTOS" encompasses everything from a tiny scheduler that fits in a few kilobytes to a full-featured OS with megs of code supporting a GUI, networking stack, file system, and more. uCLinux and Microsoft's .NET come to mind as "computer-in-drag" software that can run on an MCU.

Nevertheless, you've got to draw the line in the silicon somewhere. At least for now, when I refer to an MCU, you'll know what I'm talking about.

ARMs RACE

ARM MCUs are unique among all of the others by virtue of their "open architecture" with a bandwagon of licensees, parts, and third-party supporters that's rolling so fast it's hard to keep up. Give NXP Semiconductors (under their former guise as Philips Semiconductors) credit for being first to shatter the myth that 32-bit chips are only for "high-end" designs (see Photo 1). A chip like their LPC2101 may have a 32-bit ARM7 core, but otherwise, it is remarkably streamlined with just 8 KB of flash memory, 2 KB of RAM, and a small 48-pin package.

Now NXP is making moves at the high end, notably via their recent

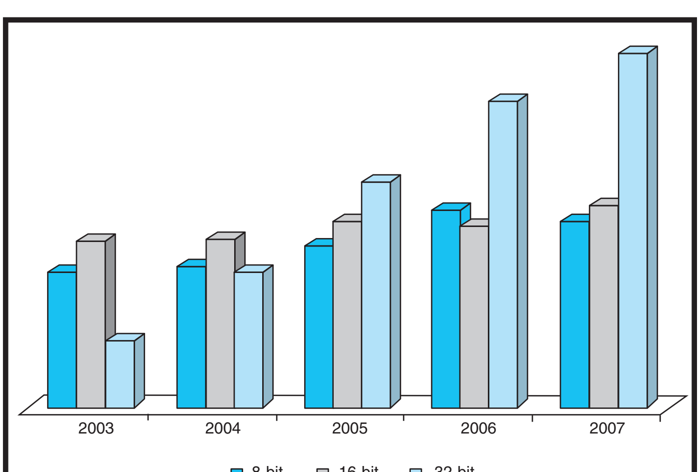


Figure 1—Development tool sales are a good leading indicator of design trends. Today, 8- and 16-bit MCUs still dominate unit shipments and indeed designs continue apace. But as this sales data from tool supplier IAR Systems Software shows, 32-bit MCUs are where tomorrow's action is.

Photo 1—Check out this cool gadget using an NXP ARM7-based LPC2119 MCU. Remember when "32 bits" was a big deal? It's still a big deal in terms of exploding growth and new applications, but this time it's a big deal that comes in a single, tiny package.

acquisition of Sharp's "Bluestreak" product line. Bluestreak parts like the ARM9-based LH7A00 bring LCD applications into the mix with a built-in display controller, frame buffer, and touchscreen interface. NXP has also rolled out their own home-grown ARM9-based LPC3000 line starting with the 90-nm, 208-MHz, 320-pin, vector floating-point LPC3180.

Per the earlier discussion, these parts are really MPUs. They don't have the flash needed for single-chip operation, but they include "computer" features (e.g., virtual memory MMU, cache, a high-speed external bus, and more) that can handle operating systems like Windows CE and Linux. But never fear, NXP is filling the high-end MCU gap with their ARM9-based LPC2900 series. These strip out the MPU features and replace them with flash memory to deliver a high-performance single-chip solution.

Atmel is another powerhouse that, like NXP, offers a full catalog of MCUs (ARM7) and MPUs (ARM9). However, unlike NXP, Atmel uses a stacked die approach for integrating memory on their MCUs. The virtue of this multi-chip-in-package strategy can be debated, but there's no doubt that it's the way to go if you want a lot of memory. Software bloat getting you down? Relief is just an AT91 MCU (with up to 2 MB of flash memory and 256 KB of SRAM) away.

The MCU business isn't for the faint of heart, and it is dominated by companies with decades of leadership. Conventional wisdom would say the idea of starting a new MCU company this late in the game is iffy at best. It's testimony to the momentum behind ARM that the introduction of their latest and greatest Cortex M3 MCU core spawned a new MCU company, namely Luminary Micro.

Like pregnancy, being "kind of" in the MCU business isn't an option. Dribbling out a part or two while watching the cash gauge bounce on "E" won't cut it. Fortunately for

Luminary, their backers appear to have the means and will to make a go of it. They shot out of the starting gate with a complete line-up of general-purpose MCUs ranging from the such-a-deal $1 LM3S101 (8 KB of flash memory, 2 KB of SRAM, 28-pin) to the LM3S1968 (256 KB of flash memory, 64 KB of SRAM, 100-pin). Now, they've virtually doubled the size of their portfolio with a second wave of parts tailored for Ethernet, CAN, and motor control applications.

Another notch in ARM's Cortex-M3 belt comes from STMicroelectronics, who have added the core to their extensive line-up of MCUs. Make no mistake that ST is a major player with a lot of weight to throw around, and it shows in the initial strength of their product offering and third-party bandwagon (see Figure 2). Chips on the table range from 32 KB of flash memory and 36 pins, to 128 KB of flash memory and 100 pins, with 512 KB and 144 pins in the works. Support comes from a who's-who of tool suppliers, including Keil (now owned by ARM), Hitex Development Tools, Raisonance, and IAR Systems, just to name a few.

SOFT PARADE

Designers have an abundance of standard MCUs to choose from for middle-of-the-road applications. At the other extreme, ultra-high-volume products may call for a custom ASIC. Filling the gap are soft-core processors running on FPGAs (i.e., a roll-your-own SoC solution).

Leaving no socket unturned, ARM is making their own soft-core moves by working with Actel to deliver ARM7 capability in flash-based FPGAs. Just recently, the duo announced a new, more streamlined Cortex-M1 core that cuts the gate count and power consumption, factors that are always a consideration with FPGAs.

For their part, Xilinx is boosting aspirations for their Microblaze core with the addition of a full-fledged Windows CE and Linux-capable MMU to the architecture. For applications that don't need the full MMU baggage but still want some protection, synthesis alternatives include basic USER/SUPERVISOR protection and a limited hardware (i.e., no virtual memory) "memory protection unit" (see Table 1).

The ability to trade-off price, power, and performance is key to the soft-core concept. For example, the Altera Nios II core comes in three "flavors" (fast, standard, and economy), giving designers the flexibility to optimize for a wide range of different applications. Flexibility also extends to implementation options with Altera and Synopsys collaborating to offer NIOS II for real ASIC designs. I guess that makes it a hard-core soft core.

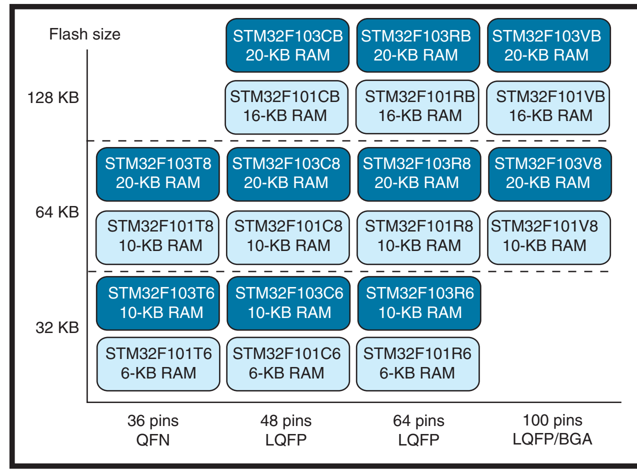


Figure 2—Right out of the gate, STMicroelectronics offers more than a dozen Cortex M3-based STM32 MCUs divided into the "Access" (i.e., 36-MHz, low-cost) and "Performance" (72 MHz with USB and CAN) lines. But that's not all. Parts with more flash memory (e.g., 512 KB) and more pins (e.g., 144) are already in the works.

"V"

The name is different and the iconic batwing logo is gone, but Freescale Semiconductor carries forward the historic embedded leadership of Motorola. Over the years, Motorola seemingly never met a 32-bit architecture they didn't love. There was 68K/Coldfire, PowerPC, ARM, M-Core, and 88K, not to mention DSPs.

Except for the 88K (lived fast, died young), these all soldier on under the Freescale banner. But Coldfire is the core that Freescale expects to do the heavy lifting when it comes to general-purpose MCU applications. While the chip name may have changed, the family resemblance of Coldfire to the venerable Motorola 68K is no coincidence. Designers have made it clear that when it comes to embedded, the old ways are usually the good (enough) ways.

The latest "V1" version of Coldfire cuts the fat to compete in the low-end "buck and change" segment, which is where growth is really exploding. For less than $2 (10K quantity), you can get a chip that delivers nearly 50 MIPS while consuming only 50 mW (e.g., 27 mA at 1.8 V). We're not talking about a stripped-down loss leader either. There's 32 KB of flash memory, 8 KB of RAM, and plenty of upscale I/O including 20-channel 12-bit ADC, two each SCI, I^2C, SPI, and up to 54 general-purpose I/Os.

Another interesting move is what Freescale calls their "controller continuum" strategy (see Figure 3). The idea is to ease upward and downward migration across their entire 8-, 16-, and 32-bit line-up. The linkage is established with bridge-the-gap parts that use different cores (e.g., one chip using an 'S08, the other a Coldfire V1) but otherwise offer exactly the same I/O, package, and pinout.

PRESCRIPTION FOR CHANGE

Did you know that Renesas Technology has shipped a billion flash MCUs? Don't apologize if you didn't. The fact is, much of the volume is concentrated in giant, but hidden, applications such as automotive. Most of their business is figuratively, not to

	Virtex-5 Resources	Spartan-3 Resources
Memory Management Unit (MMU)	910 LUTs, 1 Block RAM	1,100 LUTs, 1 Block RAM
Memory Protection Unit (MPU)	560 LUTs, 1 Block RAM	670 LUTs, 1 Block RAM
User/Supervisor Privilege	34 LUTs	38 LUTs

Table 1—With the addition of virtual-memory MMU, the Xilinx Microblaze soft core joins the ranks of Windows CE and Linux-compatible MPUs. Of course, the raison d'etre for an FPGA soft core is the flexibility to jettison baggage if you don't need it. So, simpler memory management solutions are an option.

mention literally, "under the hood."

It's true that a lot of Renesas MCUs reach the general-purpose mass market, but the success is spread across a catalog bursting with decades worth of legacy parts handed down from Hitachi and Mitsubishi. To be fair, the line-up is pretty coherent at the low end (e.g., entry-level H8s) and the high end (SuperH). It's in between where positioning gets tricky with high-line H8s and middle-of-the-road M16s and M32s all jostling for the same sockets.

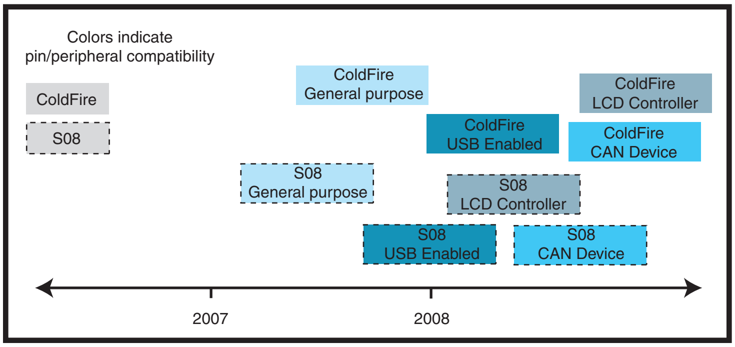


Figure 3—Freescale Semiconductor bridges the gap between 8-bit and 32-bit designs by giving otherwise pin- and I/O-compatible parts a brain transplant.

Enter the new "RX" family, which is said to stand for "Renesas Extreme." Frankly, it's also the Rx for marketing indigestion brought on by architectural overindulgence. RX is said to offer big-tent compatibility and an upgrade path for all the existing mid-range parts. For example, there's explicit "bi-Endian" support to accommodate the "big Endian" and "little Endian" parts in the current line-up (see Figure 4).

Technically, what I've seen of RX so far makes sense to me. Notably, I give Renesas credit for calling a spade a spade—or, in this case, a CISC. Most marketeers still call their chips RISCs, although at this point, virtually every distinguishing characteristic of the original concept has fallen out of favor.

Take the issue of instruction encoding, with fashion having changed faster than hemlines. The original RISC concept insisted on fixed-length 32-bit instructions. Although sellable at the time, I always thought it was a goofy idea. Spend 32 bits to increment a register, compare to zero, do a short branch? It wasn't long before folks realized the ostensible benefit of a 32-bit fixed-length instruction (slightly simpler and faster silicon) doesn't nearly justify the cost in terms of wasted code space and bus bandwidth.

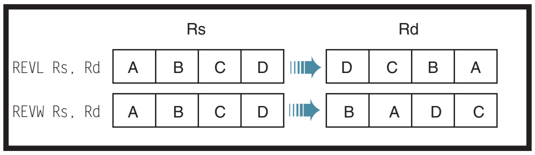


Figure 4—Providing an upgrade path for a myriad of legacy MCUs is the challenge for the forthcoming Renesas Technology RX MCU. Doing so starts with the basics, namely the ability to handle data in big or little Endian format.

So we went from fixed-length 32-bit instructions to adding "modes" (e.g., Thumb and MIPS16) that carved out a subset of leaner 16-bit instructions. More recently, we see Cortex M3 with native 16- and 32-bit instructions.

RX goes even further with a byte-variable instruction length that encodes the most frequently used instructions in the least amount of bits. That's a great idea—just like it always was.

MIPS FOR THE MASSES

The icing on the 32-bit MCU cake comes from Microchip Technology. Say hello to the MIPS-based PIC32 (see Figure 5).

This is an interesting announcement on a number of levels. From Microchip's perspective, a MIPS deal gives them a credible and expedient alternative to a "me too" ARM or "hard sell" proprietary core strategy. Meanwhile, MIPS gets to expand their presence beyond the ASIC business into the merchant MCU market. That's not just an upside for their bottom line, but a must to boost their long-term prospects for architectural staying power.

I haven't followed the MIPS architecture closely because it's pretty much been positioned as an ASIC-only option. Skimming the datasheet, I can see that from high altitude, PIC32 certainly resembles the MIPS chips of yore (e.g., R3000 and R4000). But look a little more closely and you'll find evidence of the changing times. There are clearly some "RISC versus Reality" face-offs where traditional RISC theology blinks first.

For instance, in their quest for brutal simplicity, old-timey RISCs were known for "Relegating Impossible Stuff to the Compiler." Indeed, the "MIPS" moniker is said to have stood for "Microprocessor without Interlocking Pipeline Stages." Instead of hardware interlocks, the idea was to have the compiler schedule around hazards by moving instructions, inserting NOPs, and so on. So, I have to smile now that I see in the PIC32 datasheet that "Pipeline interruptions are handled entirely in hardware."

Another example is interrupts—or rather, in the case of the original MIPS design, lack of same. As I recall, software was expected to handle nearly every aspect of interrupt processing, with the chip doing little more than stacking the PC and status word in a single-deep backup register. Want to know where the interrupt came from? Use software. Want to handle nested interrupts? Use software. Want to implement priority? Use software. By contrast, the "Rev.2" variant of the architecture used in PIC32 has an altogether modern vectored interrupt capability and even a spare register set dedicated to the cause of exception processing.

There are plenty of other tweaks. For instance, just toggling a bit on the early RISCs required a 32-bit load/mask/store sequence. Not only is this wasteful in many ways, it's also slow and subject to glitches due to the fact it takes multiple cycles (i.e., updates are "non-atomic"). PIC32 brute-forces the traditional RISC bit I/O problem away by supplementing each peripheral register with three other registers dedicated to setting, clearing, and inverting bits (see Figure 6).

THAT'S NOT ALL FOLKS

I've touched on half a dozen eminently respectable 32-bit architectures comprising hundreds of parts supplied by a top-tier list of global IC powerhouses. Could you ask for anything more? Of course, when it comes to silicon, it

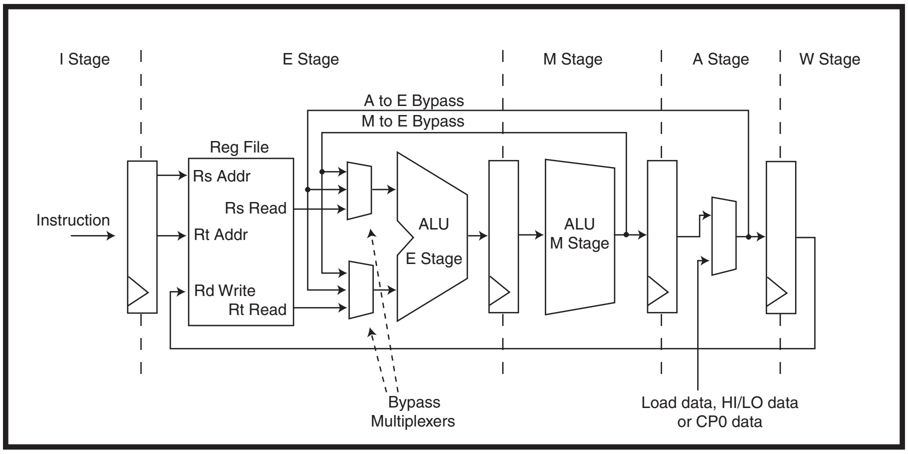


Figure 5—At the heart of the PIC32 is a powerful MIPS core with performance-oriented features, including pre-fetch cache, virtual memory MMU, and a high-speed math coprocessor. The sophisticated-for-an-MCU five-stage pipeline reduces stalls by forwarding results directly to waiting stages.

doesn't matter if you ask because you're getting it anyway.

There's a whole parallel universe of DSPs that are a must for the most signal-centric applications. However, all the DSP suppliers (e.g., Analog Devices, Freescale, and TI) have been working hard to offer chips that are more MCU-like, both in terms of hardware features (e.g., flash and I/O) and development environment ("C," RTOS, and debug). If you're up for an ASIC, there are custom cores from the likes of MIPS, ARC, and Tensilica. Then there's multicore, where you've got everything from a $10 Propeller chip to sci-fi parts with hundreds of cores today, thousands tomorrow.

What a wonderful time to be an embedded designer. Combine these new-age MCUs with other advances (e.g., sensors, wireless, and Internet), and the race for new applications is on. You've got some really strong horses to choose from. Now, it's time to ride. ■

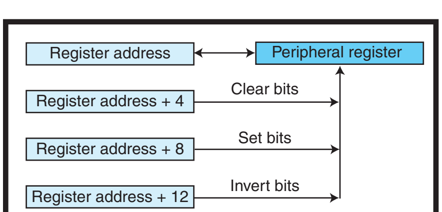


Figure 6—In the beginning, RISCs were all about reducing hardware by doing things in software. Unfortunately, that doesn't work well for control tasks such as bit I/O. The solution, as embodied in the PIC32, is simple: reduce software by doing things in hardware.

Tom Cantrell has been working on chip, board, and systems design and marketing for several years. You may reach him by e-mail at tom.cantrell@circuitcellar.com.

SOURCES

Cortex M1 Soft-core FPGA
Actel Corp.
www.actel.com

NIOS II Soft-core FPGA
Altera Corp.
www.altera.com

CAP Metal-mask programmable SoC
Atmel Corp.
www.atmel.com

Coldfire 32-bit Microcontroller
Freescale Semiconductor, Inc.
www.freescale.com

IDE, RTOS, and TCP/IP
IAR Systems Software, Inc.
www.iar.com

PIC32 Microcontroller
Microchip Technology, Inc.
www.microchip.com

LPC2900, LPC3000, and LPC3180 Microcontrollers
NXP Semiconductors
www.nxp.com

H8, RX, and SuperH Microcontrollers
Renesas Technology Corp.
www.renesas.com

STM32 Microcontroller
STMicroelectronics
www.st.com

Microblaze soft-core FPGA
Xilinx, Inc.
www.xilinx.com

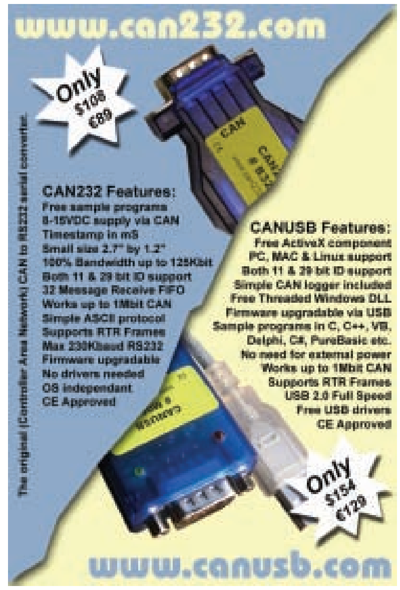
IDEA BOX

THE DIRECTORY OF PRODUCTS AND SERVICES

AD FORMAT: Advertisers must furnish digital submission sheet and digital files that meet the specifications on the digital submission sheet. **ALL TEXT AND OTHER ELEMENTS MUST FIT WITHIN A 2″ x 3″ FORMAT**. Call for current rate and deadline information. Send your disk and digital submission sheet to: IDEA BOX, Circuit Cellar, 4 Park Street, Vernon, CT 06066 or e-mail adcopy@circuitcellar.com. For more information call Shannon Barraclough at (860) 875-2199.

The Suppliers Directory at www.circuitcellar.com/suppliers_dir/
is your guide to a variety of engineering products and services.

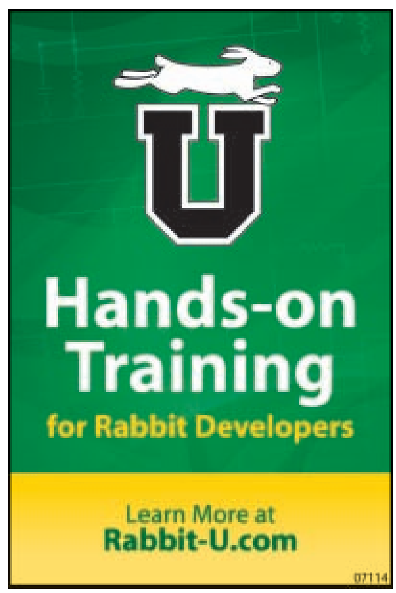
EMULATE A USB
Keyboard + Mouse + Joystick
NEW!
MODEL KE-USB108
-108 individual programmable switch inputs
-Produce long macros from a single input
-Send keystrokes,mouse or joystick actions
-Ideal handling of toggle switches
-2 Rotary Encoder inputs
-Potentiometer inputs for Joystick X,Y,Z
888-690-9080
HAGSTROMELECTRONICS.COM
11 FIDDLERS GREEN, LANSING NY 14882



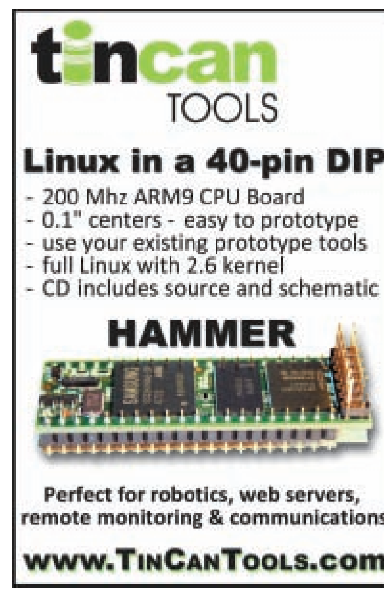
www.can232.com
Only $108 €89
The original (Controller Area Network) CAN to RS232 serial converter.
CAN232 Features:
Free sample programs
8-15VDC supply via CAN
Timestamp in mS
Small size 2.7" by 1.2"
100% Bandwidth up to 125Kbit
Both 11 & 29 bit ID support
32 Message Receive FIFO
Works up to 1Mbit CAN
Simple ASCII protocol
Supports RTR Frames
Max 230Kbaud RS232
Firmware upgradable
No drivers needed
OS independant
CE Approved
CANUSB Features:
Free ActiveX component
PC, MAC & Linux support
Both 11 & 29 bit ID support
Simple CAN logger included
Free Threaded Windows DLL
Firmware upgradable via USB
Sample programs in C, C++, VB,
Delphi, C#, PureBasic etc.
No need for external power
Works up to 1Mbit CAN
Supports RTR Frames
USB 2.0 Full Speed
Free USB drivers
CE Approved
Only $154 €129
www.canusb.com

ROBOT POWER
Extreme Motor Speed Control!
$399
Sidewinder
$79.99
Simple-H
• 14V - 50V
• Dual 150A+ Peak Bridges
• Current/Temp limiting
• R/C inputs - serial option
• Extruded Aluminum case
• 4.25" x 3.23" x 1.1"
• 5.5V - 28V
• 45A Peak H-bridge
• Current/Temp limiting
• 3- wire interface
• 2.5" x 2.25" x 0.5"
• Control with your micro
1.6"x .625"
$29.99
1.6" sq
$79.99
1.6" x 2.7"
$119.99
Powerful!
Scorpion Mini - Single 6A H-bridge - R/C interface
Scorpion HX - Dual 6A H-bridge +12A Aux - R/C
Scorpion XL - Dual 30A H-bridge - R/C or serial
www.robotpower.com
Phone: 253-843-2504 • sales@robotpower.com

Product Development
Industrial Design
Electrical & Mechanical Engineering
Prototyping and Manufacturing
Quick-turn PCB
Prototype Assembly
As little as one board
Engineering Your Path to Profit
IMET
Corporation
2014 Ford Road, Unit B • Bristol, PA 19007
Tel: 215-788-5256 • 866-435-4907
Fax: 215-788-5362
www.imetcorporation.com



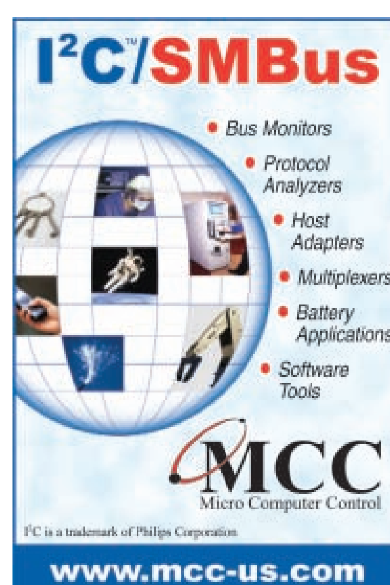
U
Hands-on
Training
for Rabbit Developers
Learn More at
Rabbit-U.com
07114



tincan
TOOLS
Linux in a 40-pin DIP
- 200 Mhz ARM9 CPU Board
- 0.1" centers - easy to prototype
- use your existing prototype tools
- full Linux with 2.6 kernel
- CD includes source and schematic
HAMMER
Perfect for robotics, web servers,
remote monitoring & communications
www.TinCanTools.com

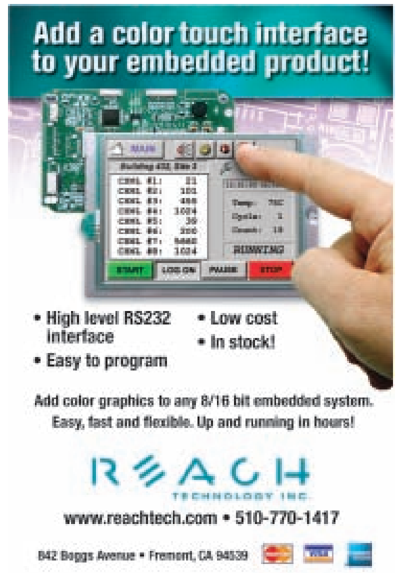
1,600+ Products
www.DesignNotes.com
Project Cases
Connectors
& Adapters
100's of
Predesigned
Electronic Kits
K8048
Test Equipment
USB PC 2 CH
60MHZ Oscilloscope
$329.99
+ free gift
PCSU1000
DesignNotes.com
Affordable Prototyping Products
1-800-957-6867

Outsource Your Next
Design To Ohio!
Product Line Strategy Consulting
Circuit Design and PCB Layout
Electromechanical Algorithm Development
Analog Design
Wireless
Simulation
Embedded Software
Rapid Prototype
We specialize
in designing
with Microchip
products
MICROCHIP
AUTHORIZED
DESIGN
PARTNER
O'Keefe Electronics Inc.
www.okeefe-electronics.com
440 647 5193
O'Keefe Electronics Inc.



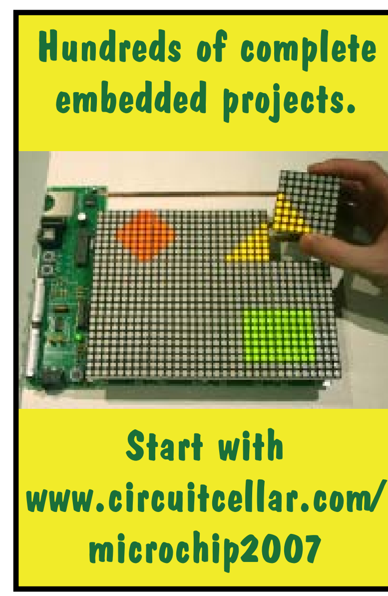
I²C/SMBus
• Bus Monitors
• Protocol Analyzers
• Host Adapters
• Multiplexers
• Battery Applications
• Software Tools
MCC
Micro Computer Control
I²C is a trademark of Philips Corporation
www.mcc-us.com


Solve complex signal acquisition problems...
• positioning & control
• environmental
• acceleration
• transients
• pressure
• vibration
• sonar
• GPS
• Linux Driver
• Guaranteed in stock
• Customization available
• 16-bit analog inputs and outputs
• Million sample FIFO eliminates interrupts
• Wide analog input and output ranges
• -40°C to +85°C Standard
www.stx104.com
Apex Embedded Systems
sales@stx104.com • 608-256-0767 x24


Add a color touch interface to your embedded product!
• High level RS232 interface
• Easy to program
• Low cost
• In stock!
Add color graphics to any 8/16 bit embedded system.
Easy, fast and flexible. Up and running in hours!
REACH
TECHNOLOGY INC.
www.reachtech.com • 510-770-1417
842 Boggs Avenue • Fremont, CA 94539


Loadstar SENSORS
Load Sensing Made Easy
Introducing iLoad Pro™
Rugged, low profile digital load/force sensors with USB or analog output.
No need for signal conditioning or data aquisition systems. Just Plug and Sense™
World's First USB Load Cell!
SAVE 10%
If you order before April 1, 2008
Use PROMO CODE: 04CC10
www.loadstarsensors.com
650.938.4282
sales@loadstarsensors.com


ANSI C
iMAGEcraft
software development tools for:
Cypress PSoC
Atmel AVR
HC08,11,12,16
TI MSP430
ARM
Global Optimizer
Full-featured 45-day demos on our website
hardware kits:
Elektronikladen - HC08 & HCS12 / 12 ref. design & starter kits
CANDIP - AVR/CAN on a chip!
ACME - FOX Boards
Ethernut - Complete AVR TCP/IP solution
high powered development, garage powered prices!
www.imagecraft.com
info@imagecraft.com
706 Colorado Ave. Palo Alto, CA 94303
(650) 493-9326 • FAX (650) 493-9329
VISA / MC / AMEX
Check / Money Order
...say 'NO' to JURASSIC PRICES!


Hundreds of complete embedded projects.
Start with www.circuitcellar.com/microchip2007

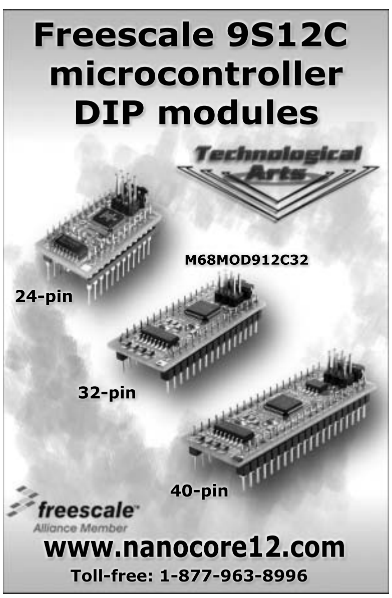

MYLYDIA, INC.
Free one time setup!
(subject to approval)
W/5 double side PCB
2 weeks
Tel: 800.695.9342
sales@mylydia.com
www.mylydia.com


H-Drive™
Host USB, Ethernet, CompactFlash, RS232, Relays, ADC, DAC, and TFT
• 4.3 x 3.3", Easy to program in C/C++
• Two USB Host Ports for USB Flash Disk, Keyboard/mouse
• 100M Base-T Ethernet, 6 RS232, I/O, 5.7" TFT support
• 16 ch 24-bit ADC, 11 ch 12-bit ADC, 8 ch. 16-bit DAC
• CompactFlash card with FAT file system support
• Solenoid drivers, and Reed Relays
start at $169 Qty 1
60+ Low Cost Controllers with ADC, DAC, Solenoid drivers, Relays, CF, LCD, DSP motion control, UARTs, 300 I/Os. Custom board/software design. Save time and money.
TERN INC.
1950 5th Street, Davis, CA 95616 USA
Tel: 530-758-0180 • Fax: 530-758-0181
www.tern.com • sales@tern.com

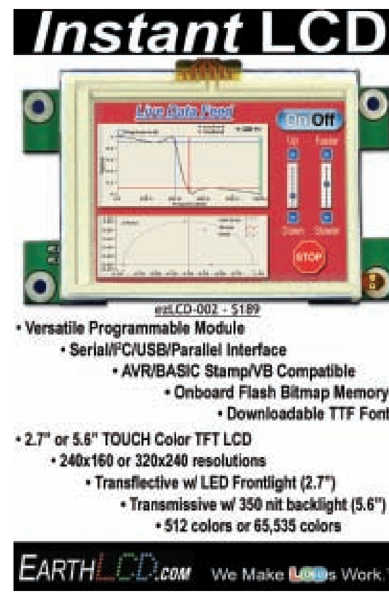

The $69 PLC
Work as Stand-Alone Ladder Logic PLC.
Or as Smart Remote I/Os of PC/ PLCs.
RS485 allows 256 units to be networked.
E10-Relay+
RS485
Opto232
RS232
RS485
E10-npn+
Incredibly Easy to Program!
Our software is used by many colleges for teaching PLCs!
Get Free Ladder Logic Simulator:
www.tri-plc.com/cci.htm
Triangle Research
Tel: 1-877-874-7527 - PLC specialist since 1993


SpectraPLUS 5.0
Audio Spectrum Analysis
Features
Sound Card based I/O
FFT sizes to 1048576pts, 1/96 Octave
Up to 24 bit, 200kHz sampling rates
3-D Surface and Spectrogram
Digital Filtering, Signal Generation
THD, IMD, SNR, Transfer Functions
DDE, Macros, Data Logging,
Vibration Analysis, Acoustic Tools
FREE 30 day trial!
www.spectraplus.com
PHS
Pioneer Hill Software
360 697-3472 voice
pioneer@telebyte.com


CAN-4-USB FX
USB to CAN Interface
USB 2.0 Hi-Speed 480Mbps!
2008 marks our 12th year of selling CAN interfaces in over 30 countries!
New Low Price!
$199
Zanthic Technologies Inc.
403-878-2202
www.zanthic.com


Freescale 9S12C microcontroller DIP modules
Technological Arts
M68MOD912C32
24-pin
32-pin
40-pin
freescale
Alliance Member
www.nanocore12.com
Toll-free: 1-877-963-8996


HI-TECH
SOFTWARE
SAVE
50%
ANSI C COMPILERS
supporting Microchip
PICmicro® MCUs and DSCs
For more details go to:
www.htsoft.com/portal/CC_50
PICmicro is a registered trademark of Microchip Technology Inc.

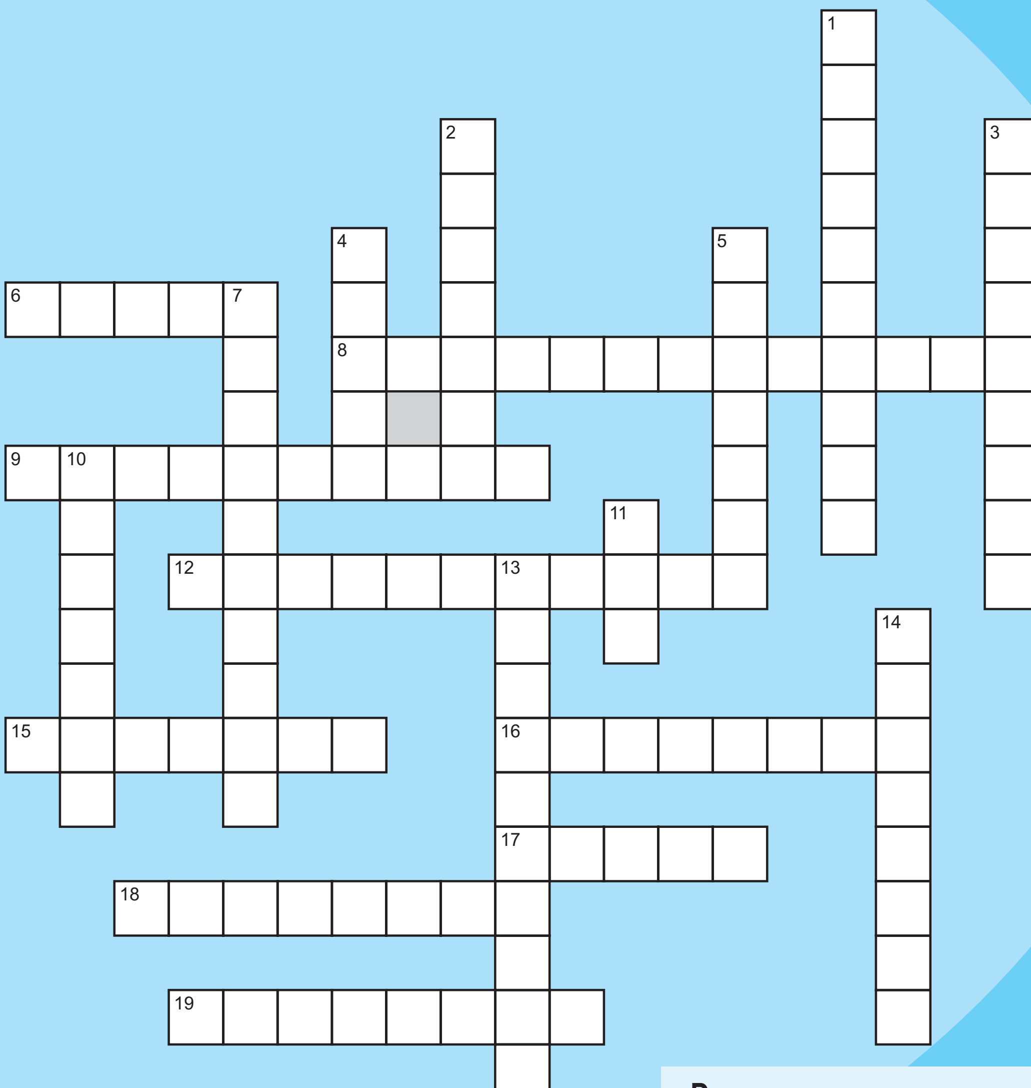

Instant LCD
ezLCD-002 - $189
• Versatile Programmable Module
• Serial/I2C/USB/Parallel Interface
• AVR/BASIC Stamp/VB Compatible
• Onboard Flash Bitmap Memory
• Downloadable TTF Fonts
• 2.7" or 5.6" TOUCH Color TFT LCD
• 240x160 or 320x240 resolutions
• Transflective w/ LED Frontlight (2.7")
• Transmissive w/ 350 nit backlight (5.6")
• 512 colors or 65,535 colors
EARTHLCD.COM
We Make LCDs Work.™


CROSSWARE
Tools for Embedded Development
ARM®
8051
COLDFIRE®
Three quality tool suites
❑ C/C++*
❑ Code Wizards
❑ Debugging
❑ Simulation
❑ Support
* Embedded C++ for ARM and ColdFire only
Advanced software tools since 1984
www.crossware.com
360-812-2397


FAT 12/16/32
FILE SYSTEM
• DOS/Windows Compatible
• USB Flash Disk
• USB Floppy & Hard Disk
• SD/MMC
• CompactFlash, ATA/IDE
• DiskOnChip
• NAND & NOR Flash
• 10KB RAM / 25KB ROM typical
• Low Cost, No Royalty
• Full Source Code
www.smxrtos.com
Micro Digital Inc
RTOS Innovators
800.366.2491 sales@smxrtos.com


CUSTOM MEMBRANE
KEYBOARDS / SWITCHES
•1 TO 2 WEEKS TURNAROUND
•VERY COMPETITIVE PRICING
•Ex.: (5) 4-switch keyboards for $395.00
•PCB backed switches
•Custom metal backplates/assemblies
•Electronic assemblies/graphic overlays
•Electronic file transfer capabilities
Picofab Inc.
4780B Blvd. Henri-Bourassa
Charlesbourg, Quebec, Canada G1H 3A7
Tel: (418) 622-5298 • Fax: (418) 622-9996
Email: sales@picofab.net

CROSSWORD



Across

6. A short description on a web site
8. An instrument used to measure acceleration on a sensor
9. Created the Playfair cipher in 1854
12. A PC's 'Net-connected location
15. Shaped like a spiral or coil
16. A mobile electronic device that fits in your hand
17. Smoke produced by dysfunctional circuits
18. Think: General Electric Research Lab, ductile tungsten, 1927 IEEE Edison Medal
19. Project Genie began at this University of California campus in 1964

Down

1. Unofficial rules of conduct that regulate interaction and behavior on the Internet
2. The main material used in most semiconductors
3. Legal protection that enables the creator of original material to restrict the use and reproduction of his/her work
4. A receiver/transmitter that decodes parallel data from a CPU into serial bits
5. A datatype that has two values: "1" and "0"
7. This occurs when the operation of an entire system is restricted by one component
10. A headphone and microphone system used to communicate over a telephone
11. This type of "iron" refers to a fast supercomputer
13. The ancient Greek engineer, mathematician, and astronomer who was killed in Syracuse in 212 BC
14. Fi

The answers are available at www.circuitcellar.com/crossword.

INDEX OF ADVERTISERS

The Index of Advertisers with links to their web sites is located at www.circuitcellar.com under the current issue.

Preview of April Issue 213
Theme: Embedded Programming

ATA I/O: Use ATA Interfaces for General-Purpose I/O Applications

Simple Hardware Development

Programmable Power: Build a Simple USB DAC

Do-It-Yourself Motion-Controlled Gaming (Part 2): Manipulate 3-D Graphics

Video Stamp: A Video Display for Debugging Programs

Embedded Linux Development (Part 1): A Virtual Linux Environment on a Windows System

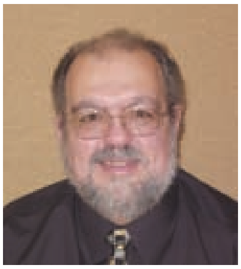
---

THE DARKER SIDE **Low Power Techniques:** Build Better Energy-Saving Electronic Systems

ABOVE THE GROUND PLANE **TRIAC Control Meets An Inductive Load**

FROM THE BENCH **Digital Touch:** A Potentiometer with No Moving Parts

SILICON UPDATE **STM32**



PRIORITY INTERRUPT

by Steve Ciarcia, Founder and Editorial Director

What Happened to Performance?

It seems like I've been spending a lot more editorial space ranting these days. I think the reason is that I feel more at the mercy of ridiculous engineering than I used to. There was a time when if I didn't like how something worked, I'd just go re-design or customize it, but with today's complex processors and million-line operating systems, re-engineering commercial items is pretty much impossible. Basically, we're at the mercy of whatever corporate profit-inspired crap they want to shove our way. Do I sound bitter?

Whether your embedded expertise involves software or hardware, our professional lives these days mandate the use of a PC and there aren't a whole lot of options. Choices regarding system configuration and processor speeds are still ruled by our wallet, as they should be in a capitalistic economy, but the sad choice of operating systems is a terrible statement of monopoly politics.

I have three computers (two Sony VAIOs and a Toshiba) that I use on a regular basis. One is a large desktop and the other two are large and small laptops. All of them are about three years old and use Windows XP Pro. For the most part, they have been bug-free and I haven't had any problems installing software for the many programs and devices that I like to use. However, like everything technical these days, evolution continues. Certainly, now that we have dual and quad processors galore, I could expect to see a remarkable increase in performance in a couple of new top-of-the-line VAIOs with Microsoft's latest and greatest multi-processor operating system—Vista.

I'm glad I wasn't the first one on my block to try this because it would have been a disaster. But in order to get a better handle on reality versus all of the Microsoft bashing, I bought a dual-processor Gateway laptop with Vista Home Premium to see the results of three more years of Moore's Law in action. In my opinion, Vista is a sexy, graphic-intensive operating system designed to move us all into the world of flashy video, Internet-centric living, and more Microsoft-provided software services and advertising. (Sounds a lot like Google doesn't it?) My single-processor 2.39-GHz P4 XP seems faster than a 2.6-GHz Core 2 Duo with twice the RAM running Vista. And, there seems to be very little driver support for the small-company peripherals and software packages that I am currently using under XP. (It should have been a clue when I did a Google search on "Vista sucks" and got 150,000 results using exactly those words.) ;-)

At this point, the decision is to forget upgrading (a concept that if it catches on should strike panic in every PC manufacturer), switch to a Mac or Ubuntu (unfortunately not enough driver support here either), or buy new hardware and just install XP Pro instead of Vista (easier said than done).

I hope PC manufactures get the message, but they need to know they have been led down the path by Microsoft. Perhaps like many of you, I'm voting with my wallet. I'm not buying any new PC hardware until they sell it to me without Vista or in a configuration that I can easily install XP alone. For example, we needed new desktops for everyone at the office this year. To the detriment of all the other PC manufacturers who only sell Vista, the solution was getting them from Dell because Dell would configure them preloaded with XP.

Unfortunately, I don't think Dell has the laptops I want, and solving my personal PC requirements is turning into a whole 'nother bag of worms. Unless I order some expensive exotic gaming machine where they already know it has to be XP or a way off-brand PC, I don't see any laptops from the "regular guys" that aren't preloaded with Vista. (And who's the idiot that thought up four different versions of Vista to further confuse the market?) Unlike a desktop where we can configure the hardware to accommodate the available drivers and do a clean XP install, prepackaged laptops are black boxes with few configuration options. The only option for most VAIO, HP, Toshiba, etc. laptop owners is to try to install XP as a second disk partition. The good news is that this indeed gives you back the use of XP, but apparently Vista never goes away, and doing this has its own unique set of bugs. Forget that.

Still, hardware is hardware, isn't it? Logic suggests that we should be able to simply buy a new HP or Gateway laptop, wipe Vista off the machine, and then install XP. Unfortunately, the reality is that most retailers are selling a lot of this new hardware as "Vista-only" machines and internal peripheral XP drivers aren't necessarily available for that specific laptop. The result is that it may require numerous calls to the manufacturer (how's your Chinese?) to obtain XP drivers. Yes, you can install XP, but you might find that the DVD player doesn't work because you have the wrong driver.

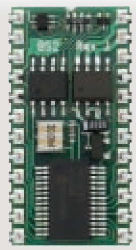
So, right now I'm sitting here with three computers I'd like to upgrade and about $6,000 I'm not spending on any manufacturer until someone sells me a new laptop with XP Pro on it. I have a list of favorite brands, but I'm open to suggestions. If you've spent more time researching this problem and have some answers, please e-mail me and tell me where to look. In the meantime, I'll trust that Vista Service Pack 1 has helped mitigate some of the performance issues, but until then, I guess Google's content scanning will make this editorial 150,001 the next time someone searches that phrase.

steve.ciarcia@circuitcellar.com

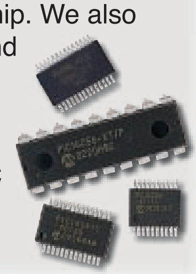
Steve

Ready to build in quantity? We stock our PBASIC interpreter chips, SX microcontrollers, and the Propeller chip in rails/trays with production-volume pricing. Visit www.parallax.com for information on package types and stock availability.

BASIC Stamp

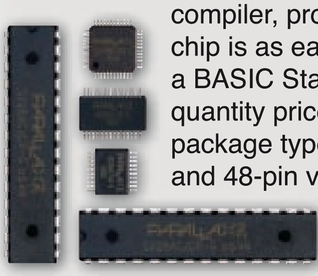


Give your product a BASIC Stamp brain. Build a BASIC Stamp microcontroller on your PCB with our PBASIC Interpreter Chip. We also stock compatible resonators and EEPROM. Eleven PBASIC interpreter chip program types are available, from the BASIC Stamp 1 to the powerful BASIC Stamp 2p family.



SX Chips

The SX features an 8-bit timer, analog comparator, watchdog timer, interrupt capability, and 75+ MIPS. With our free SX/B compiler, programming an SX chip is as easy as programming a BASIC Stamp, with a great quantity price. Available in 5 package types including 20, 28, and 48-pin varieties.



Propeller Chip

Simplify your embedded designs with the Parallax Propeller. This high-speed parallel-processing microcontroller has eight 32-bit processors that can perform truly simultaneous tasks, independently or cooperatively, with deterministic timing. Programming is surprisingly straightforward with native high-level Spin and assembly. Free resources include our community forums and full source code examples in our online object library (http://obex.parallax.com). Super-fun for lots of stuff. Available in 40-pin DIP and 44-pin LQFP and QFN packages.

2008 Propeller Design Contest
$4,000 in cash and prizes!

Calling all programmers! Sign up to enter your project in the Propeller Design Contest. See www.parallax.com/go/propellercontest for details, rules and restrictions. Last day to request a project number is September 01, 2008.

To place an order visit www.parallax.com or call our Sales Department toll-free at 888-512-1024 (Monday-Friday, 7 a.m. - 5 p.m., PST).

PARALLAX INC

www.parallax.com